Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

CRT-2003-16

A CUTTING-PLANE ALGORITHM BASED ON CUTSET INEQUALITIES FOR MULTICOMMODITY CAPACITATED FIXED CHARGE NETWORK DESIGN

by

Mervat Chouman

Teodor Gabriel Crainic

Bernard Gendron

July 2003

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone : (514) 343-7575 / Télécopieur : (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

HEC ÉCOLE POLYTECHNIQUE MONTRÉAL Université de Montréal

PUBLICATION
CRT-2003-16

A CUTTING-PLANE ALGORITHM BASED ON CUTSET INEQUALITIES FOR MULTICOMMODITY CAPACITATED FIXED CHARGE NETWORK DESIGN

by

Mervat Chouman[1]

Teodor Gabriel Crainic[2]

Bernard Gendron[1]

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR).

[1] Département d'informatique et de recherche opérationnelle and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 (mervat@crt.umontreal.ca, bernard@crt.umontreal.ca)

[2] Département de management et technologie, Université du Québec à Montréal, et Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 (theo@crt.umontreal.ca)

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2003

103-07111/

Abstract

The multicommodity capacitated fixed charge network design problem is a well-known NP-hard problem which arises in a wide variety of applications, most notably in transportation and telecommunications. In this paper, we propose to improve the mixed integer programming (MIP) formulation of the problem by using a polyhedral approach. We present a cutting-plane algorithm which incorporates valid inequalities (VI) based on cutsets of the network. We identify and study, in the field of cutset type inequalities, three families of VI: the well-known cover inequalities, the minimum cardinality inequalities, and the network cutset inequalities. We develop efficient separation heuristics and lifting procedures, as well as a cutset generation algorithm. We present computational results on a large set of instances of various characteristics. Our results show that the cutting-plane algorithm achieves better lower bounds with competitive computational effort when compared to the results obtained when solving a large-scale formulation with the state-of-the-art MIP software package CPLEX.

Key words : multicommodity capacitated fixed-charge network design, polyhedral approach, cutting-plane algorithm, valid inequalities, separation, lifting.

Résumé

Le problème de conception de réseaux multiproduits avec coûts fixes et capacités est un problème NP-difficile apparaissant dans une grande variété d'applications, notamment en transport et en télécommunications. Dans cet article, nous proposons d'améliorer la formulation de programmation en nombres entiers (PNE) du problème en utilisant une approche polyédrale. Nous présentons un algorithme de coupes qui intègre des inégalités valides basées sur des coupes du réseau. Parmi ces inégalités, nous identifions et étudions trois familles d'inégalités: les inégalités de couverture, les inégalités de cardinalité minimale et les inégalités de coupes du réseau. Nous développons des heuristiques de séparation et des procédures d'élévation séquentielle, de même qu'un algorithme de génération de coupes du réseau. Nous présentons des résultats expérimentaux obtenus sur un grand nombre de problèmes ayant différentes caractéristiques. Nos résultats montrent que l'algorithme de coupes obtient de meilleures bornes dans des temps de calcul compétitifs lorsqu'on le compare aux résultats obtenus avec une formulation de grande taille résolue au moyen du logiciel de PNE CPLEX.

Mots-clés : problème de conception de réseaux multiproduits avec coûts fixes et capacités, approche polyédrale, algorithme de coupes, inégalités valides, séparation, élévation séquentielle.

1 Introduction

Network design formulations are used to model a wide variety of applications, most notably in the fields of transportation and telecommunications (Magnanti and Wong 1984, Minoux 1989, Crainic 1999). These formulations are often characterized as follows: given a network with arc capacities, it is required to send flows in order to satisfy, at minimum cost, known demands between origin-destination pairs. In doing so, one pays a price not only for transporting flows, but also for using arcs, in terms of fixed costs. The goal of the resulting multicommodity capacitated fixed charge network design problem (MCND) is to determine the optimal amounts of flows to be transported and the arcs to be used.

The MCND is modeled as a mixed integer program (MIP) where continuous variables represent the flows and 0-1 variables are used to model the design decisions (the arcs to be used). The MCND is NP-hard since it contains as a special case the uncapacitated fixed charge network design problem, which is NP-hard as well (Magnanti and Wong 1984). In addition, considerable algorithmic challenges are imposed when solving realistically-sized problem instances. These challenges are not only due to the large size of real applications but also to the trade-off between variable and fixed costs, as well as to the competition among commodities due to the limited capacity on the arcs.

Branch-and-bound (B&B) algorithms based on linear programming (LP) relaxations are the most common tools to solve MIP problems. An effective B&B implementation for the MCND requires to improve the lower bound provided by the LP relaxation, which generally yields a poor approximation of the MIP convex hull of solutions (Gendron, Crainic, and Frangioni 1998). Gendron and Crainic (1994) proposed to improve the lower bounds by adding the so-called *strong inequalities* (presented in Section 2), which are redundant for the MIP formulation but not for its LP relaxation. Unfortunately, solving the resulting strong LP relaxation requires considerable computational time since the formulation is very large and frequently exhibits degeneracy. This seriously impacts the overall performance of the B&B algorithm.

Alternative relaxation approaches have been devised for solving the MCND, in particular Lagrangian-based procedures (Gendron and Crainic 1994, 1996, Holmberg and Yuan 2000, Gendron, Crainic, and Frangioni 1998, Crainic, Frangioni, and Gendron 2002, and Sellmann, Kliewer, and Koberstein 2002). Several Lagrangian relaxations have been used in these procedures, and all of them yield the same theoretical lower bound as the strong LP relaxation (Gendron and Crainic 1994). Heuristic strategies have also been proposed for computing feasible solutions (Crainic, Gendreau, and Farvolden 2000, Ghamlouche, Crainic, and Gendreau 2001, 2002, Crainic, Gendron, and Hernu 2002, Crainic and Gendreau 2002). To assess the quality of the solutions produced by these heuristics, optimal solutions must be known. However, to this date, even tight lower bounds cannot be computed in a reasonable amount of time for a large set of instances.

Tightening up the formulation of the problem is thus of crucial importance for the de-

velopment of efficient solution methods. This might be accomplished using the polyhedral approach, which attempts to improve the formulation by adding *valid inequalities* (or *VI*) in the form of linear constraints. This is usually performed within the framework of simplex-based cutting-plane methods. The polyhedral approach has proven effective to solve network design problems related to the MCND, such as the *network loading problem* (Magnanti, Mirchandani and Vachani 1993, 1995, Barahona 1996, Bienstock and Günlük 1996, Bienstock *et al.* 1998, Günlük 1999, Gabrel, Knippel, and Minoux 1999, Atamtürk 2002a, 2002b), and the *capacitated facility location problem* (Leung and Magnanti 1989, Aardal, Pochet and Wolsey 1995, Aardal 1998). To the best of our knowledge, however, no systematic study of the polyhedral structure of the MCND has been performed yet.

Several VI derived for particular structures of the MCND are well-known, especially the *cover* and *flow cover* inequalities (Nemhauser and Wolsey 1988). These inequalities, among others, are incorporated in branch-and-cut (B&C) algorithms implemented in standard MIP software packages, such as MPSARX (Van Roy and Wolsey 1987), MINTO (Nemhauser, Savelsbergh and Sigismondi 1994), Xpress-MP (2002) and CPLEX (2002). Chouman, Crainic and Gendron (2001) have observed that the B&C method of CPLEX (version 7.1, 2002) cannot solve large size instances of the problem in a reasonable amount of time. With more than 10 hours of CPU time allowed for each instance on a Sun Entreprise 10000. the authors have noticed that CPLEX could generate flow cover inequalities, but was unable to generate any cover inequality.

In this paper, we present a cutting-plane procedure which incorporates VI based on *cutsets* of the network. These inequalities are based on the following idea: in any feasible solution, the capacity of the arcs of any cutset that cuts off some origins from their destinations must be sufficient to flow the demand across the cutset. We are motivated by previous studies that prove the effectiveness of such inequalities for solving problems related to the MCND, in particular the network loading problem. We identify and study, in the field of cutset type inequalities, three families of VI: the well-known *cover inequalities*, the *minimum cardinality inequalities*, and the *network cutset inequalities*. The last one is, to the best of our knowledge, a new family of VI, introduced for the first time. The study of cover inequalities is motivated by the fact that the general implementation of these inequalities in CPLEX was unable to detect them. To evaluate the performance of the proposed cutting-plane procedure, we perform a computational study on a large set of test problems studied in the literature.

The main contribution of this paper is the development of computationally efficient separation and lifting procedures for these families of VI. We have also developed an efficient heuristic to identify cutsets of the network on which cuts can be generated. Our computational results show that the three families of VI are useful for the MCND. In addition, when compared to solving the strong LP relaxation with CPLEX, our cutting-plane procedure improves both the lower bounds and the time needed for their computation.

This paper is organized as follows. In the next section, we present a MIP formulation of the MCND and some basic VI. Sections 3 and 4 are dedicated to the study of two cutset

type inequalities, the cover inequalities and the minimum cardinality inequalities, respectively, and to their relationship. The network cutset inequalities are presented in Section 5. Section 6 describes the cutset generation algorithm and the cutting-plane procedure. Computational results are presented in Section 7. We conclude this paper by proposing future research related to this work.

2 Formulation and Basic Valid Inequalities

The multicommodity capacitated fixed charge network design problem (MCND) is characterized using the following notation. Given a directed graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs, and a set of commodities (or origin-destination pairs) K to be routed according to a known demand d^k for each commodity k, the problem is to satisfy the demand at minimum cost. The cost consists of the sum of transportation costs and fixed design costs, the latter being charged whenever an arc is used. The transportation cost per unit of commodity k on arc (i, j) is denoted $c_{ij}^k \geq 0$, while the fixed design cost for arc (i, j) is denoted $f_{ij} \geq 0$. An origin $O(k)$ and a destination $D(k)$ are associated to each commodity k. We introduce continuous flow variables x_{ij}^k, which reflect the amount of flow on each arc (i, j) for each commodity k, and 0-1 design variables y_{ij}, which indicate if arc (i, j) is used or not. With this notation, we write the MIP formulation of the MCND as follows:

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij}, \tag{1}$$

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = \begin{cases} d^k, & \text{if } i = O(k), \\ -d^k, & \text{if } i = D(k), \\ 0, & \text{otherwise,} \end{cases} \quad \forall\, i \in N, \forall\, k \in K, \tag{2}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \quad \forall\, (i,j) \in A, \tag{3}$$

$$x_{ij}^k \geq 0, \quad \forall\, (i,j) \in A, \forall\, k \in K, \tag{4}$$

$$y_{ij} \in \{0,1\}, \quad \forall\, (i,j) \in A, \tag{5}$$

where $N_i^+ = \{j \in N | (i,j) \in A\}$ and $N_i^- = \{j \in N | (j,i) \in A\}$.

Relations (2) correspond to flow conservation constraints for each node and each commodity. Relations (3) represent capacity constraints for each arc. They also link together flow and design variables by forbidding any flow to pass through an arc that is not chosen as part of the design.

The LP relaxation is obtained by replacing the integrality constraints (5) by

$$0 \leq y_{ij} \leq 1, \quad \forall (i,j) \in A.$$

Gendron, Crainic, and Frangioni (1998) show that the lower bounds derived from this LP relaxation are generally weak. In order to improve these lower bounds, provided by the

so-called *weak LP relaxation*, the following VI, called *strong inequalities*, can be added to obtain the *strong LP relaxation*:

$$x_{ij}^k \leq d^k y_{ij}, \quad \forall (i,j) \in A, k \in K. \tag{6}$$

Adding the strong inequalities to the model significantly improves the quality of the LP lower bounds (Gendron, Crainic, and Frangioni 1998). In order to derive other useful VI for the MCND, we are interested in particular structures that might be hidden in relaxations of the model.

By combining the capacity constraints (3) and the flow conservation equations (2), we obtain the following constraints:

$$\sum_{j \in N_i^+} u_{ij} y_{ij} \geq \sum_{k \in K | i = O(k)} d^k, \ \forall i \in N. \tag{7}$$

In order to generalize inequality (7), let $S \subset N$ be any non empty subset of N and its

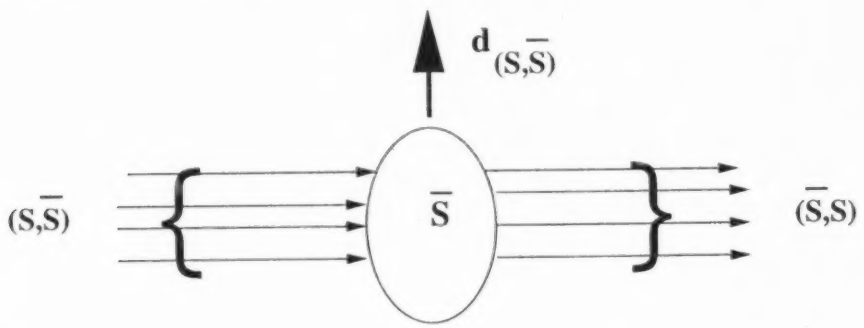

Figure 1: Cutset structure

complement $\bar{S} = N \backslash S$, and let $K(S, \bar{S}) \subseteq K$ be the non empty set of commodities having their origin in S and their destination in $\bar{S}$. Then, we have the inequality

$$\sum_{(i,j) \in (S,\bar{S})} u_{ij} y_{ij} \geq d_{(S,\bar{S})}, \tag{8}$$

where $d_{(S,\bar{S})} = \sum_{k \in K(S,\bar{S})} d^k > 0$ and $(S, \bar{S})$ is the set of arcs that connect a node in S to a node in $\bar{S}$ (a *cutset*). This is the well-known cutset inequality which states that the capacity used on the arcs of the cutset $(S, \bar{S})$ must be sufficient to satisfy the demand across the cutset. Consequently, the cutset inequality is a VI for the MCND for any $S \subset N$. Although the cutset inequality is redundant for the LP relaxation of the MCND, some VI derived from it might improve the LP relaxation.

We can derive useful VI from the cutset structure by using two different ideas. The first idea consists in complementing the y variables (replacing y_{ij} by $1 - y_{ij}$) in the cutset inequality to obtain the knapsack structure given by:

$$\sum_{(i,j) \in (S,\bar{S})} u_{ij} y_{ij} \leq \sum_{(i,j) \in (S,\bar{S})} u_{ij} - d_{(S,\bar{S})}.$$

Then, the well-known *cover inequalities* and the *minimum cardinality inequalities*, derived from this knapsack structure, are VI for any cutset, and thus for the MCND. Note that these two families of VI are not redundant in the LP relaxation of the MCND since they are based on the integrality of the y variables. The second idea, used to derive the *network cutset inequalities*, is based on the insertion of the flow variables x into the cutset inequalities.

These families of VI are described in the next three sections. We assume through these sections that a cutset $(S, \bar{S})$ is given. Heuristics used to generate such cutsets are presented in Section 6.

3 Cover Inequalities

Consider the set P_S induced by a given cutset $(S, \bar{S})$,

$$P_S = \{y \in \{0,1\}^m : \sum_{(i,j)\in(S,\bar{S})} u_{ij} y_{ij} \geq d_{(S,\bar{S})}\},$$

where $m = |(S, \bar{S})|$. As mentioned above, by complementing the y variables in P_S, one obtains the 0-1 knapsack set. Then, the cover inequalities, developed independently by Balas (1975), Hammer, Johnson, and Peled (1975) and Wolsey (1975), constitute an interesting family of VI for P_S. We now adapt to P_S the standard definitions related to cover inequalities.

Definition 1: A set $C \subseteq (S, \bar{S})$ is a cover if the total capacity of arcs in $(S, \bar{S})\backslash C$ does not cover the demand: $\sum_{(i,j)\in(S,\bar{S})\backslash C} u_{ij} < d_{(S,\bar{S})}$.

Definition 2: A cover $C \subseteq (S, \bar{S})$ is minimal if it is sufficient to open any arc in C to cover the demand: $\sum_{(i,j)\in(S,\bar{S})\backslash C} u_{ij} + u_{pq} \geq d_{(S,\bar{S})}, \forall (p,q) \in C$.

For every cover $C \subseteq (S, \bar{S})$, the *cover inequality* (CI) defined as

$$\sum_{(i,j)\in C} y_{ij} \geq 1 \tag{9}$$

is valid for P_S, for any subset S. Thus, it is valid for the MCND. The basic idea behind this inequality is that one has to open at least one arc from the set C in order to meet the demand. In addition, it has been proven (Nemhauser and Wolsey 1988) that if C is a minimal cover, then the CI is a facet of

$$conv(P_S \cap \{y \in \{0,1\}^m : y_{ij} = 1, \ \forall (i,j) \in (S, \bar{S})\backslash C\}).$$

However, the CI (9) is not a facet of $conv(P_S)$ in general and it has to be strengthened using the lifting procedure (see Section 3.2 below). In order to generate a violated CI, given that the current LP solution is fractional, one has to determine a cover C by solving the separation problem for a given cutset $(S, \bar{S})$.

3.1 Separation

Let $\bar{y}$ be a fractional optimal solution for some LP relaxation of the MCND. The separation problem of the CI consists in finding the most violated cover C, if there is one, by solving the following optimization problem:

$$\begin{array}{rcl} Z_{sep} & = & \min \quad \sum_{(i,j)\in(S,\bar{S})} Z_{ij}\bar{y}_{ij}, \\ & & \sum_{(i,j)\in(S,\bar{S})} Z_{ij}u_{ij} > \sum_{(i,j)\in(S,\bar{S})} u_{ij} - d_{(S,\bar{S})}, \\ & & Z_{ij} \in \{0,1\} \ ,\forall (i,j) \in (S,\bar{S}), \end{array}$$

where Z is the characteristic vector of the set C ($Z_{ij} = 1$ then $(i,j) \in C$). Let Z_{sep} be the optimal objective value. If $Z_{sep} < 1$, then the constructed cover C identifies the most violated cover inequality. Otherwise, the current fractional solution $\bar{y}$ does not violate any cover inequality.

This is a 0-1 knapsack problem. Even though there exists a pseudo-polynomial dynamic programming algorithm for solving exactly the 0-1 knapsack problem (Martello and Toth 1990), it is still time consuming especially because such problems must be solved repeatedly. We have implemented a faster heuristic approach that considers the arcs in non-decreasing order of $\bar{y}_{ij}$, instead of $\frac{\bar{y}_{ij}}{u_{ij}}$, as would be performed by the classical greedy heuristic for the 0-1 knapsack problem. Ties are broken by considering the arcs in non-increasing order of their capacity. Once a cover is obtained with this heuristic, it is easy to extract a minimal cover from it, by removing some of the arcs from the cover until the condition in Definition 2 is satisfied. The basic idea of this heuristic is to try to exclude as much as possible from the set C the arcs with large $\bar{y}_{ij}$, in order to increase the chance of finding a violated inequality (i.e., $\sum_{(i,j)\in C} \bar{y}_{ij} < 1$). The same heuristic has been used by Gu, Nemhauser, and Savelsbergh (1998, 1999) in their extensive study of cover inequalities. Once the cover C is constructed, the induced inequality might be strengthened by the lifting procedure. Note that, even if the identified cover inequality is not violated, we might find a violated one through the lifting procedure (see Example 1 below).

3.2 Lifting

The principle of lifting is to extend a given VI in a higher dimensional space by including the variables that do not appear in the inequality. By doing so, we associate with each of these variables a so-called lifting coefficient. Lifting the CI can be performed using two different strategies. The first one consists in lifting down all variables in $(S,\bar{S})\backslash C$ by seeking lifting coefficients, $\gamma_{ij} \geq 0$, such that the lifted cover inequality

$$\sum_{(i,j)\in(S,\bar{S})\backslash C} \gamma_{ij}y_{ij} + \sum_{(i,j)\in C} y_{ij} \geq 1 + \sum_{(i,j)\in(S,\bar{S})\backslash C} \gamma_{ij}$$

is a VI (see Example 1 below). This type of lifting contributes to the violation since $\gamma_{ij}y_{ij} \leq \gamma_{ij}$. Moreover, γ_{ij} can be chosen so that this inequality is facet-defining for $conv(P_S)$

when C is a minimal cover (Wolsey 1998).

We have implemented and tested this lifting strategy on a large set of instances. The experimental results show that this strategy is not useful for our instances. In particular, we have noted that many arcs $(i,j) \in C$ with a large capacity have a fractional $\bar{y}_{ij}$ value close to zero at the current LP solution. In this case, if the corresponding cover inequality is not violated, this lifting strategy will not, in general, produce a violated inequality. We will illustrate this situation in Example 1 below.

Considering this observation, we present a second lifting strategy, which, to the best of our knowledge, is introduced for the first time (see our remarks below, as well as Example 2, for a comparison with the lifting strategy proposed by Gu, Nemhauser, and Savelsbergh 1998). The basic idea is to determine, *a priori*, two subsets C_1 (the open arcs) and C_0 (the closed arcs) in $(S, \bar{S})$ that satisfy the condition

$$\sum_{(i,j)\in(S,\bar{S})\setminus(C_1\cup C_0)} u_{ij} \geq d_{(S,\bar{S})} - \sum_{(i,j)\in C_1} u_{ij} > 0.$$

We then define the restricted set $P_{S_{(C_1,C_0)}}$ induced by C_1 and C_0 as

$$P_{S_{(C_1,C_0)}} = \{y \in \{0,1\}^m : \sum_{(i,j)\in(S,\bar{S})\setminus(C_1\cup C_0)} u_{ij}y_{ij} \geq d_{(S,\bar{S})} - \sum_{(i,j)\in C_1} u_{ij}\}.$$

We call *restricted CI* a cover inequality derived from this restricted set, with its corresponding cover denoted C. Since this inequality is restricted to open arcs in C_1 and closed arcs in C_0, lifting (down for the variables in C_1 and up for the variables in C_0) is necessary to ensure global validity. In addition, by lifting down the variables in $(S, \bar{S})\setminus(C_1 \cup C_0 \cup C)$, as in the first lifting strategy, we obtain the *lifted cover inequality* (LCI):

$$\sum_{(i,j)\in(S,\bar{S})\setminus C} \gamma_{ij}y_{ij} + \sum_{(i,j)\in C} y_{ij} \geq 1 + \sum_{(i,j)\in(S,\bar{S})\setminus(C\cup C_0)} \gamma_{ij}. \tag{10}$$

Again, it is possible to choose the values for the lifting coefficients so that this inequality is facet-defining for $conv(P_S)$, when C is a minimal cover with respect to the restricted set. Note also that, lifting down the variables in C_1 contributes to the violation of the inequality since $\gamma_{ij}y_{ij} \leq \gamma_{ij}$. However, lifting up the variables in C_0 has a negative impact on the violation in the sense that, an inequality that appeared violated prior to this lifting step might become satisfied after. This might happen if some variables in C_0 have positive values ($\bar{y}_{ij} > 0$) at the current LP solution. Thus, the arcs in C_0 must have small values at the current solution ($\bar{y}_{ij}$ close to 0).

The choice of the sets C_1 and C_0 is of crucial importance for the performance of this lifting strategy. We propose the following procedure, called **OpenCloseArcs**, which uses the variables T and D representing, respectively, the residual capacity ($\sum_{(i,j)\in(S,\bar{S})\setminus(C_1\cup C_0)} u_{ij}$) and the residual demand ($d_{(S,\bar{S})} - \sum_{(i,j)\in C_1} u_{ij}$). The procedure attempts to close an arc (i,j) with a small value $\bar{y}_{ij}$ (as measured by a parameter ϵ_0) and such that the residual capacity ($T - u_{ij}$) still covers the residual demand D. Similarly, the procedure attempts to open an

arc (i,j) with a large value $\bar{y}_{ij}$ (as measured by a parameter ϵ_1) and such that there is still some residual demand to cover ($D - u_{ij} > 0$). The outline of this procedure is summarized next.

Procedure **OpenCloseArcs:**

- $T \leftarrow \sum_{(i,j)\in(S,\bar{S})} u_{ij}$, $D \leftarrow d_{(S,\bar{S})}$.
- For each arc $(i,j) \in (S,\bar{S})$ (in arbitrary order):
  - If ($\bar{y}_{ij} < \epsilon_0$) and ($T - u_{ij} \geq D$) then
    - Add (i,j) to C_0;
    - Close (i,j) by setting $T \leftarrow T - u_{ij}$;
  - If ($\bar{y}_{ij} > \epsilon_1$) and ($D - u_{ij} > 0$) then
    - Add (i,j) to C_1;
    - Open (i,j) by setting $D \leftarrow D - u_{ij}$ and $T \leftarrow T - u_{ij}$.

Once the sets C_1 and C_0 are obtained, the lifting procedure is applied sequentially, meaning that the variables are lifted one after the other in some predetermined order. At a given step of the lifting procedure, suppose we are lifting variable y_{rt}. Then, let $L = \{(i,j) \in (S,\bar{S})\backslash C,\ y_{ij}$ has been lifted$\}$, $\bar{C}_1 = (S,\bar{S})\backslash(C \cup C_0)$, $\gamma_{ij} = 1, \forall (i,j) \in C$ and $\bar{d} = d_{(S,\bar{S})} - \sum_{(i,j)\in\bar{C}_1\backslash L} u_{ij}$. To *lift down* variable $y_{rt} \in (S,\bar{S})\backslash(C \cup C_0)$, we have to solve the following 0-1 knapsack problem:

$$\begin{array}{rcll} Z_{opt} = \min & \sum_{(i,j)\in C\cup L} \gamma_{ij} y_{ij}, & & \\ & \sum_{(i,j)\in C\cup L} u_{ij} y_{ij} & \geq & \bar{d} + u_{rt}, \\ & y_{ij} & \in & \{0,1\},\ \forall (i,j) \in C \cup L. \end{array}$$

If this problem is feasible, the lifting coefficient is given by $\gamma_{rt} = Z_{opt} - 1 - \sum_{(i,j)\in L\backslash C_0} \gamma_{ij}$; otherwise, we set $\gamma_{rt} = \sum_{(i,j)\in C\cup L} \gamma_{ij} - \sum_{(i,j)\in L\backslash C_0} \gamma_{ij}$ (this is equivalent to $y_{rt} = 1$, which follows from the infeasibility of the problem). To *lift up* variable $y_{rt} \in C_0$, we have to solve the following 0-1 knapsack problem:

$$\begin{array}{rcll} Z_{opt} = \min & \sum_{(i,j)\in C\cup L} \gamma_{ij} y_{ij}, & & \\ & \sum_{(i,j)\in C\cup L} u_{ij} y_{ij} & \geq & \bar{d} - u_{rt}, \\ & y_{ij} & \in & \{0,1\},\ \forall (i,j) \in C \cup L. \end{array}$$

If this problem is feasible, the lifting coefficient is given by $\gamma_{rt} = 1 + \sum_{(i,j)\in L\backslash C_0} \gamma_{ij} - Z_{opt}$; otherwise, we set $\gamma_{rt} = \sum_{(i,j)\in L\backslash C_0} \gamma_{ij} - \sum_{(i,j)\in C\cup L} \gamma_{ij}$ (this is equivalent to $y_{rt} = 0$).

Since the lifting coefficients have small values, the 0-1 knapsack problems are solved efficiently using a dynamic programming algorithm. Also, the quality of the resulting inequality depends on the lifting order. In particular, lifting down the variables in $(S,\bar{S})\backslash C \cup C_0$ must be accomplished before lifting up the variables in C_0 (since the former contribute to the

violation). Also, when lifting down the variables in $(S, \bar{S})\backslash C \cup C_0$, those with fractional values must be lifted first, in non-decreasing order of their current value. Ties are broken by considering first the arcs in non-increasing order of their capacity. When lifting up the variables in C_0, we do the exact opposite. Example 1 illustrates the two lifting strategies just described.

Example 1: Consider the following cutset inequality:

$$13y_1 + 7y_2 + 6y_3 + 4y_4 + 3y_5 + 11y_6 + 22y_7 \geq 22,$$

and let $\bar{y} = (0, 0.5, 0.5, 0.6, 0.7, 1, 0)$ be the current LP solution, restricted to the arcs in the cutset. Note that the inequality is satisfied at equality by this solution.

First strategy: Generate a CI and then lift down all variables in $(S, \bar{S})\backslash C$*:* To generate the most violated cover we have to solve the following separation problem:

$$\begin{aligned} \min \quad & 0.5z_2 + 0.5z_3 + 0.6z_4 + 0.7z_5 + z_6, \\ & 13z_1 + 7z_2 + 6z_3 + 4z_4 + 3z_5 + 11z_6 + 22z_7 > 44, \\ & z_i \in \{0,1\}, \ \forall i = 1, ..., 7. \end{aligned}$$

Using the heuristic described above, we find the following CI:

$$y_1 + y_2 + y_3 + y_7 \geq 1,$$

which is not violated. Note that this inequality corresponds to a minimal cover. Lifting down y_4, y_5 and y_6 in that order, according to the strategy described above, gives the same inequality since we find $\gamma_i = 0, i = 4, 5, 6$. In this case, the lifting does not improve the inequality, although it identifies a facet of $conv(P_S)$.

Second strategy: Let $\epsilon_1 = 0.6$ *and* $\epsilon_0 = 0.1$*, so that* $C_1 = \{5, 6\}$ *and* $C_0 = \{1, 7\}$*; generate an LCI by first lifting down all variables in* $S\backslash C \cup C_0$ *and then lifting up all variables in* C_0*:* The resulting restricted set $P_{S_{(C_1,C_0)}}$ is defined by the inequality

$$7y_2 + 6y_3 + 4y_4 \geq 8.$$

Applying the separation heuristic, we find the restricted CI induced by C_1 and C_0:

$$y_2 + y_3 \geq 1.$$

Note that this inequality corresponds to a minimal cover with respect to the restricted set. Lifting down y_4, y_5 and y_6 and then lifting up y_1 and y_7 in that order, according to the strategy described above, gives the following LCI:

$$3y_1 + y_2 + y_3 + y_4 + 2y_6 + 4y_7 \geq 4,$$

which is valid for P_S but violated by 0.4 by the current LP solution. This LCI also defines a facet of $conv(P_S)$.

Gu, Nemhauser, and Savelsbergh (1998) propose a similar lifting strategy, where the sets C_1 and C_0 are derived from the variables having integer values at the current LP solution. This approach can be simulated with Procedure **OpenCloseArcs** by using ϵ_0 arbitrarily close to 0 and ϵ_1 arbitrarily close to 1. One might ask if there are cases where this strategy does not provide a violated inequality, while using a larger value for ϵ_0 and a smaller value for ϵ_1 will identify a cut. The following example illustrates this situation.

Example 2: Consider the following cutset inequality:

$$3y_1 + 3y_2 + 3y_3 + 4y_4 + 5y_5 \geq 10.$$

Let $\bar{y} = (0.1, 0.9, 0.9, 0.1, 0.9)$ be the current LP solution. Since no variable assumes an integer value, the strategy described above gives $C_1 = C_0 = \emptyset$. Using the separation heuristic, we derive the minimal cover $C = \{4, 5\}$, since we include successively arcs 4, 1, and 5, thus obtaining a cover, and then we remove arc 1 to ensure the cover is minimal. The corresponding inequality is:

$$y_4 + y_5 \geq 1.$$

When we apply the lifting procedure, we lift down successively y_1, y_2, and y_3, but the inequality is not improved, since we obtain $\gamma_i = 0, i = 1, 2, 3$. We note that this inequality is not violated by $\bar{y}$.

Let $\epsilon_1 = 0.99$ and $\epsilon_0 = 0.2$, so that $C_1 = \emptyset$ and $C_0 = \{1, 4\}$. The resulting restricted set $P_{S_{(C_1,C_0)}}$ is defined by the inequality

$$3y_2 + 3y_3 + 5y_5 \geq 10.$$

Applying the separation heuristic, we find the restricted CI induced by C_1 and C_0:

$$y_5 \geq 1.$$

Lifting down y_2 and y_3 and then lifting up y_1 and y_4 in that order gives the following LCI:

$$y_1 + y_2 + y_3 + y_4 + y_5 \geq 3,$$

which is valid for P_S but violated by 0.1 by the current LP solution $\bar{y}$.

4 Minimum Cardinality Inequalities

By a simple rounding argument, we can derive from (8) the following inequality:

$$\sum_{(i,j)\in(S,\bar{S})} y_{ij} \geq \left\lceil \frac{d_{(S,\bar{S})}}{u_{max}} \right\rceil \tag{11}$$

where $u_{max} = \max_{(i,j)\in(S,\bar{S})} u_{ij}$. When the magnitude of the capacities vary widely, (11) provides a weak inequality as illustrated in Example 3.

Example 3: Consider the following constraint:

$$15y_1 + 5y_2 + 3y_3 + 2y_4 + y_5 + 5y_6 \geq 30.$$

For this example $\left\lceil \frac{d_{(S,\bar{S})}}{u_{max}} \right\rceil = 2$, while, in fact, we have to open at least 5 arcs ($y_1 = y_2 = y_3 = y_4 = y_6 = 1$) to satisfy the constraint.

In order to obtain a stronger inequality having the same left-hand side as (11), we define the *minimum cardinality inequality* (MCI):

$$\sum_{(i,j)\in(S,\bar{S})} y_{ij} \geq l_S,$$

where $l_S = \max\{h : \sum_{t=1,...,h} u_{ij_{(t)}} < d_{(S,\bar{S})}\} + 1$ and $u_{ij_{(t)}} \geq u_{ij_{(t+1)}}, t = 1, ..., m-1$. In this inequality, l_S is the least number of arcs in $(S, \bar{S})$ that must be open in every feasible solution. This inequality has been used by Martello and Toth (1997) to strengthen relaxation bounds for the 0-1 knapsack problem.

Note that, the separation problem for the MCI is trivial: for a given cutset $(S, \bar{S})$, l_S is computed efficiently by a greedy algorithm. However, the MCI is not a strong inequality in general and it has to be strengthened using a lifting procedure.

4.1 Lifting

The lifting procedure for the MCI is similar to that for the LCI. We first apply procedure **OpenCloseArcs** to determine the sets C_1 and C_0. We then derive a *restricted MCI* valid for the restricted set $P_{S_{(C_1,C_0)}}$, and we denote by $l_{S_{(C_1,C_0)}}$ the least number of arcs that must be open in the restricted set. To obtain a VI for P_S, it is then necessary to lift down the variables in C_1 and to lift up the variables in C_0. Consequently, we seek lifting coefficients γ_{ij} such that the *lifted minimum cardinality inequality (LMCI)*,

$$\sum_{(i,j)\in C_1\cup C_0} \gamma_{ij} y_{ij} + \sum_{(i,j)\in(S,\bar{S})\setminus C_1\cup C_0} y_{ij} \geq l_{S_{(C_1,C_0)}} + \sum_{(i,j)\in C_1} \gamma_{ij}, \tag{12}$$

is valid for P_S.

The lifting procedure is similar to that used to generate the LCI. More specifically, we introduce the notation $\tilde{C} = (S, \bar{S})\setminus(C_1 \cup C_0)$, $\gamma_{ij} = 1, \forall (i,j) \in \tilde{C}$ and $\bar{d} = d_{(S,\bar{S})} - \sum_{(i,j)\in C_1\setminus L} u_{ij}$. Then, to *lift down* variable $y_{rt} \in (S, \bar{S})\setminus C_0$, we have to solve the following 0-1 knapsack problem:

$$\begin{array}{rcll} Z_{opt} = & \min & \sum_{(i,j)\in\tilde{C}\cup L} \gamma_{ij} y_{ij}, & \\ & & \sum_{(i,j)\in\tilde{C}\cup L} u_{ij} y_{ij} & \geq \quad \bar{d} + u_{rt}, \\ & & y_{ij} & \in \{0,1\}, \ \forall (i,j) \in C \cup L. \end{array}$$

If this problem is feasible, the lifting coefficient is given by $\gamma_{rt} = Z_{opt} - l_{S_{(C_1,C_0)}} - \sum_{(i,j)\in L\setminus C_0} \gamma_{ij}$; otherwise, we set $\gamma_{rt} = \sum_{(i,j)\in\widetilde{C}\cup L} \gamma_{ij} - l_{S_{(C_1,C_0)}} - \sum_{(i,j)\in L\setminus C_0} \gamma_{ij} + 1$ (this is equivalent to $y_{rt} = 1$). To *lift up* variable $y_{rt} \in C_0$, we have to solve the following 0-1 knapsack problem:

$$\begin{array}{rcll} Z_{opt} = & \min & \sum_{(i,j)\in\widetilde{C}\cup L} \gamma_{ij} y_{ij}, & \\ & & \sum_{(i,j)\in\widetilde{C}\cup L} u_{ij} y_{ij} & \geq \quad \bar{d} - u_{rt}, \\ & & y_{ij} & \in \{0,1\}, \ \forall (i,j) \in C \cup L. \end{array}$$

If this problem is feasible, the lifting coefficient is given by $\gamma_{rt} = l_{S_{(C_1,C_0)}} + \sum_{(i,j)\in L\setminus C_0} \gamma_{ij} - Z_{opt}$; otherwise, we set $\gamma_{rt} = l_{S_{(C_1,C_0)}} + \sum_{(i,j)\in L\setminus C_0} \gamma_{ij} - \sum_{(i,j)\in\widetilde{C}\cup L} \gamma_{ij} - 1$ (this is equivalent to $y_{rt} = 0$).

In the same way as for the LCI, the 0-1 knapsack problems are solved using a dynamic programming algorithm. Also, lifting down the variables in C_1 must be accomplished before lifting up the variables in C_0. When lifting down the variables in C_1, those with fractional values must be lifted first, in non-decreasing order of their current value. Ties are broken by considering first the arcs in non-increasing order of their capacity. When lifting up the variables in C_0, we do the opposite. Example 4 illustrates the impact of this lifting strategy.

Example 4: As in Example 1, consider the following cutset inequality:

$$13y_1 + 7y_2 + 6y_3 + 4y_4 + 3y_5 + 11y_6 + 22y_7 \geq 22$$

and let $\bar{y} = (0, 0.5, 0.5, 0.6, 0.7, 1, 0)$ be the current LP solution. In every feasible solution, we have to open at least one arc (y_7) to satisfy the demand. The MCI is thus

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 \geq 1.$$

This inequality is not violated by the current LP solution.

To generate the LMCI, let $\epsilon_1 = 0.6$ and $\epsilon_0 = 0.1$, as in Example 1. With Procedure **OpenCloseArcs**, we find $C_1 = \{5, 6\}$ and $C_0 = \{1, 7\}$. The cutset inequality restricted to $P_{S_{(C_1,C_0)}}$ is

$$7y_2 + 6y_3 + 4y_4 \geq 8.$$

The least number of arcs to open in $P_{S_{(C_1,C_0)}}$ is $l_{S_{(C_1,C_0)}} = 2$, so the restricted MCI is

$$y_2 + y_3 + y_4 \geq 2.$$

To obtain an inequality that is valid for P_S, we lift down y_5 and y_6 and then we lift up y_1 and y_7, according to the lifting order strategy described above. We then obtain the following LMCI:

$$3y_1 + y_2 + y_3 + y_4 + 2y_6 + 4y_7 \geq 4.$$

Note that, we have generated the same inequality as in Example 1. It is thus natural to look more deeply into the relationship between LCI and LMCI. In particular, given that

we apply the same lifting order strategy with the same sets C_1 and C_0, one might ask if there are cases where one family of VI provides a cut, while the other does not. The following example illustrates these situations.

Example 5: Consider the same cutset inequality as in Example 2:

$$3y_1 + 3y_2 + 3y_3 + 4y_4 + 5y_5 \geq 10.$$

Let $\bar{y}^1 = (0.1, 0.9, 0.9, 0.1, 0.9)$ and $\bar{y}^2 = (0.1, 0.99, 0.99, 0.1, 0.85)$ be two LP solutions. It is easy to verify that both solutions satisfy this cutset inequality. We will assume that $C_1 = C_0 = \emptyset$, a result that would be obtained if Procedure **OpenCloseArcs** is performed with $\epsilon_1 = 1$ and $\epsilon_0 = 0$.

When we apply the CI separation heuristic for any of the two LP solutions, we derive the minimal cover $C = \{4, 5\}$, as in Example 2. The corresponding CI is:

$$y_4 + y_5 \geq 1.$$

When we apply the lifting procedure, this inequality is not improved, as shown in Example 2. We note that this inequality is not violated by $\bar{y}^1$, but is violated by $\bar{y}^2$.

The least number of arcs to open in P_S is $l_S = 3$. Thus, the MCI is:

$$y_1 + y_2 + y_3 + y_4 + y_5 \geq 3.$$

This inequality is violated by $\bar{y}^1$, but not by $\bar{y}^2$.

As shown by this example, with the same lifting strategy, it is possible to generate two different inequalities, one being violated by the current LP solution, while the other is not. Motivated by this observation, we have incorporated the LCI and the LMCI into our cutting-plane procedure.

5 Network Cutset Inequalities

Any feasible solution of the MCND is defined by the continuous flow variables, x_{ij}^k, and the 0-1 design variables, y_{ij}. Yet, the VI presented in the previous sections involve only the design variables. In order to derive VI that involve the two types of variables, let $L \subseteq K$, $x_{ij}^L = \sum_{k \in L} x_{ij}^k$, $b_{ij}^L = \min\{u_{ij}, \sum_{k \in L} d^k\}$ and $d_{(S,\bar{S})}^L = \sum_{k \in K(S,\bar{S}) \cap L} d^k$. Further assume that for any arc $(i, j) \in A$, any set $L \subseteq K$ is partitioned into two subsets L_{ij}^1 and L_{ij}^0. We characterize any feasible solution of the MCND by Proposition 1.

Proposition 1: Let $L \subseteq K$, $(r, t) \in (S, \bar{S})$ and $C_2 \subseteq (\bar{S}, S)$. Then,

$$x_{rt}^L \leq \Big(\sum_{(j,i) \in C_2} b_{ji}^{L_{ji}^1} + d_{(S,\bar{S})}^L \Big) y_{rt} + \sum_{(j,i) \in C_2} x_{ji}^{L_{ji}^0} + \sum_{(j,i) \in (\bar{S},S) \setminus C_2} x_{ji}^L, \tag{13}$$

is a VI for the MCND.

Proof: Let (x, y) be a feasible solution of the MCND. Then,

- if $x^L_{rt} = 0$, inequality (13) is satisfied;
- if $x^L_{rt} > 0$,

$$\begin{aligned}
x^L_{rt} &\leq \sum_{(i,j)\in(S,\bar{S})} x^L_{ij}, \\
&= d^L_{(S,\bar{S})} + \sum_{(j,i)\in(\bar{S},S)} x^L_{ji}, \\
&\quad \text{(flow conservation)} \\
&= d^L_{(S,\bar{S})} + \sum_{(j,i)\in C_2} x^L_{ji} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji}, \\
&= d^L_{(S,\bar{S})} + \sum_{(j,i)\in C_2} x^{L^1_{ji}}_{ji} + \sum_{(j,i)\in C_2} x^{L^0_{ji}}_{ji} + \sum_{(j,i)\in(S,\bar{S})\setminus C_2} x^L_{ji}, \\
&\leq d^L_{(S,\bar{S})} + \sum_{(j,i)\in C_2} b^{L^1_{ji}}_{ji} + \sum_{(j,i)\in C_2} x^{L^0_{ji}}_{ji} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji}, \\
&\quad \text{(capacity constraint)} \\
&= (\sum_{(j,i)\in C_2} b^{L^1_{ji}}_{ji} + d^L_{(S,\bar{S})})\, y_{rt} + \sum_{(j,i)\in C_2} x^{K^0_{ji}}_{ji} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji}, \\
&\quad \text{(since } y_{rt} = 1\text{)}
\end{aligned}$$

Inequality (13) might be violated by a fractional LP solution, as shown in the following example.

Example 6: Consider a set $S \subset N$ such that $(S, \bar{S}) = \{1, 2\}$, $(\bar{S}, S) = \{3, 4\}$, and $d_{(S,\bar{S})} = 2$. Let the capacities of the arcs 1, 2, 3, and 4 be given by 8, 4, 4, and 1, respectively. Also, assume that there is only one commodity and let $\bar{x} = (1, 2, 0, 1)$ and $\bar{y} = (0.125, 0.5, 0, 1)$ be the current LP solution. Given that $C_2 = \{4\}$, we obtain the following inequalities:

$$x_1 - x_3 - 3y_1 \leq 0,$$

derived for arc 1 and violated by 0.625;

$$x_2 - x_3 - 3y_2 \leq 0,$$

derived for arc 2 and violated by 0.5.

Based on Proposition 1, we introduce the *single-arc network cutset inequality (SNCI)*, defined by Proposition 2 and illustrated in Figure 2.

Proposition 2: Let $L \subseteq K$, $(r, t) \in (S, \bar{S})$, $C_1 \subseteq (S, \bar{S})\setminus\{(r, t)\}$ and $C_2 \subseteq (\bar{S}, S)$. Then,

$$\sum_{(i,j)\in C_1} x^{L^1_{ij}}_{ij} + x^L_{rt} \leq (\sum_{(j,i)\in C_2} b^{L^1_{ji}}_{ji} + d^L_{(S,\bar{S})})y_{rt} + \sum_{(j,i)\in C_2} x^{L^0_{ji}}_{ji} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji} + (1 - y_{rt}) \sum_{(i,j)\in C_1} b^{L^1_{ij}}_{ij} \quad (14)$$

is a VI for the MCND.

Proof: Let (x, y) be a feasible solution of the MCND. Then,

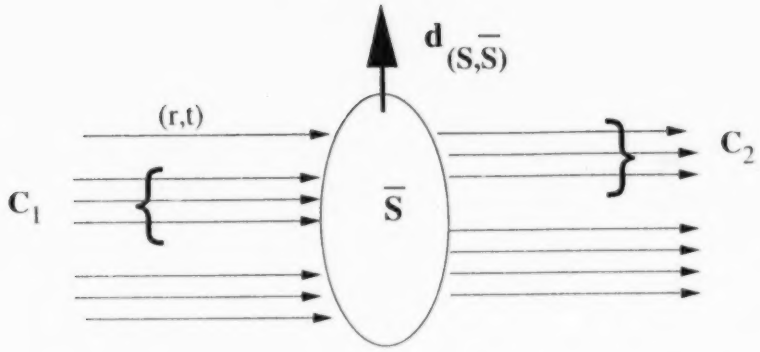

Figure 2: Network cutset structure

- if $y_{rt} = 0$, then $x^L_{rt} = 0$ and (14) is valid since $\sum_{(i,j)\in C_1} x_{ij}^{L^1_{ij}} \leq \sum_{(i,j)\in C_1} b_{ij}^{L^1_{ij}}$;

- if $y_{rt} = 1$,

$$\begin{aligned}
\sum_{(i,j)\in C_1} x_{ij}^{L^1_{ij}} + x^L_{rt} &\leq \sum_{(j,i)\in C_2} x^L_{ji} + d^L_{(S,\bar{S})} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji}, \\
&\qquad \text{(flow conservation)} \\
&\leq \sum_{(j,i)\in C_2} b_{ji}^{L^1_{ji}} + d^L_{(S,\bar{S})} + \sum_{(j,i)\in C_2} x_{ji}^{L^0_{ji}} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji}, \\
&\qquad \text{(capacity constraint)} \\
&= (\sum_{(j,i)\in C_2} b_{ji}^{L^1_{ji}} + d^L_{(S,\bar{S})}) y_{rt} + \sum_{(j,i)\in C_2} x_{ji}^{L^0_{ji}} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji}. \\
&\qquad \text{(since } y_{rt} = 1)
\end{aligned}$$

When $\sum_{(i,j)\in C_1} x_{ij}^{L^1_{ij}} > (1 - y_{rt}) \sum_{(i,j)\in C_1} b_{ij}^{L^1_{ij}}$, the SNCI (14) is stronger than inequality (13). The next example illustrates this situation.

Example 7: Consider a set $S \subset N$ such that $(S, \bar{S}) = \{1, 2\}$, $(\bar{S}, S) = \{3, 4\}$, and $d_{(S,\bar{S})} = 1$. Let the capacities of the arcs 1, 2, 3, and 4 be given by 2, 3, 3, and 4, respectively. Also, assume that there is only one commodity and let $\bar{x} = (2, 1, 2, 0)$ and $\bar{y} = (1, 1/3, 2/3, 0)$ be the current LP solution. Given that $C_2 = \{3\}$, generating the SNCI (14) for arc 2, with $C_1 = \{1\}$, gives the inequality

$$x_1 + x_2 - x_4 - 2y_2 \leq 2,$$

which is violated by 1/3. But, generating the inequality (13) for arc 2 gives

$$x_2 - x_4 - 4y_2 \leq 0,$$

which is not violated.

The SNCI considers each arc $(i, j) \in (S, \bar{S})$ separately. We can also consider multiple arcs in a single inequality, called the *multiple arc network cutset inequality (MNCI)*, introduced in the following proposition.

Proposition 3: Let $L \subseteq K$, $C \subseteq (S,\bar{S}), C_1 \subseteq (S,\bar{S})$ $(C \cap C_1 = \emptyset), C_2 \subseteq (\bar{S},S)$ and $Y_C = \max_{(r,t)\in C} y_{rt}$. Then,

$$\sum_{(i,j)\in C_1} x_{ij}^{L_{ij}^1} + \sum_{(r,t)\in C} x_{rt}^{L} \leq (\sum_{(j,i)\in C_2} b_{ji}^{L_{ji}^1} + d_{(S,\bar{S})}^{L})Y_C + \sum_{(j,i)\in C_2} x_{ji}^{L_{ji}^0} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x_{ji}^{L} + (1-Y_C)\sum_{(i,j)\in C_1} b_{ij}^{L_{ij}^1}, \quad (15)$$

is a VI for the MCND.

Proof: Let (x,y) be a feasible solution of the MCND. Then,

- if $y_{rt} = 0$, $\forall (r,t) \in C$, then $Y_C = 0$ and the MNCI is valid since

$$\sum_{(i,j)\in C_1} x_{ij}^{L_{ij}^1} \leq \sum_{(i,j)\in C_1} b_{ij}^{L_{ij}^1};$$

- if there exists $(r,t) \in C$ such that $y_{rt} = 1$, then $Y_C = 1$ and

$$\begin{aligned}
\textstyle\sum_{(i,j)\in C_1} x_{ij}^{L_{ij}^1} + \sum_{(r,t)\in C} x_{rt}^{L} \;&\leq\; \textstyle\sum_{(j,i)\in C_2} x_{ji}^{L} + d_{(S,\bar{S})}^{L} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x_{ji}^{L}, \\
&\qquad \text{(flow conservation)} \\
&\leq\; \textstyle\sum_{(j,i)\in C_2} b_{ji}^{L_{ji}^1} + d_{(S,\bar{S})}^{L} + \sum_{(j,i)\in C_2} x_{ji}^{L_{ji}^0} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x_{ji}^{L}, \\
&\qquad \text{(capacity constraint)} \\
&=\; (\textstyle\sum_{(j,i)\in C_2} b_{ji}^{L_{ji}^1} + d_{(S,\bar{S})}^{L})Y_C + \sum_{(j,i)\in C_2} x_{ji}^{L_{ji}^0} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x_{ji}^{L}. \\
&\qquad \text{(since } Y_C = 1)
\end{aligned}$$

5.1 Separation

Given a set $S \subset N$ and an arc $(r,t) \in (S,\bar{S})$, the separation problem for the SNCI consists in finding the sets L, $L_{ij}^1 \subseteq L$ for each arc (i,j), $C_1 \subseteq (S,\bar{S})$ and $C_2 \subseteq (\bar{S},S)$ such that the violation of the VI with respect to the current LP solution, $(\bar{x},\bar{y})$, is maximal. Clearly, it is optimal to set

$$L = \{k \in K : \bar{x}_{rt}^k > 0\},$$

$$L_{ij}^1 = \{k \in L : \bar{x}_{ij}^k > 0\}, \quad \forall (i,j) \in (\bar{S},S) \cup (S,\bar{S}),$$

$$C_1 = \{(i,j) \in (S,\bar{S}) \setminus \{(r,t)\} : \bar{x}_{ij}^{L_{ij}^1} > (1-\bar{y}_{rt}) b_{ij}^{L_{ij}^1}\},$$

$$C_2 = \{(j,i) \in (\bar{S},S) : b_{ji}^{L_{ji}^1} \bar{y}_{rt} < \bar{x}_{ji}^{L_{ji}^1}\}.$$

Note that, with this choice of C_1, the SNCI (14) always dominates inequality (13).

The separation problem for the MNCI is more difficult. However, once we have determined C, we can efficiently construct the other sets as in the generation of the SNCI. Based on this observation, we have devised a procedure, **Gen-MNCI**, which accepts as input the values of two user-supplied parameters, $\bar{Y}$ and δ.

Procedure **Gen-MNCI:**

2- Let $C \leftarrow \{(i,j) \in (\bar{S},S) |\ \bar{y}_{ij} \leq Y_C\}$ and readjust Y_C by setting $Y_C \leftarrow \max_{(i,j)\in C} \bar{y}_{ij}$.

3- Let $L = \{k \in L : \exists (r,t) \in C\ \bar{x}^k_{rt} > 0\}$ and $L^1_{ij} = \{k \in K : \bar{x}^k_{ij} > 0\}$, $\forall (i,j) \in (\bar{S},S) \cup (S,\bar{S})$; construct C_1 and C_2 as follows:

- $C_1 \leftarrow \{(i,j) \in (S,\bar{S}) |\ b_{ij}^{L^1_{ij}}(1 - Y_C) < \bar{x}_{ij}^{L^1_{ij}}\}$,
- $C_2 \leftarrow \{(j,i) \in (\bar{S},S) |\ b_{ji}^{L^1_{ji}} Y_C < \bar{x}_{ji}^{L^1_{ji}}\}$.

4- If the MNCI is violated, then add it to the formulation, along with the inequalities

- $Y_C \geq y_{rt},\ \forall (r,t) \in C$,
- $Y_C \leq \sum_{(r,t)\in C} y_{rt}$,
- $0 \leq Y_C \leq 1$.

5- Otherwise, give Y_C a smaller value, $Y_C \leftarrow Y_C - \delta$;

- if $Y_C > 0$, return to 2,
- else STOP.

Note that, if a violated MNCI is generated at step 4, one has to add the violated inequality, but also a new variable Y_C and a set of linear inequalities equivalent to $Y_C = \max_{(r,t)\in C} y_{rt}$.

6 Algorithm Description

Recall that, all VI presented and studied in the previous sections are based on the assumption that a cutset $(S,\bar{S})$ is given. In the following subsections, we address the issues of how to determine "good" cutsets and how to use these cutsets within a cutting-plane procedure. In Section 6.1, we describe heuristics to generate cutsets, while in Section 6.2, we present the outline of the cutting-plane procedure.

6.1 Cutset Generation Algorithm

Given that each cutset might allow to generate several types of cuts (LCI, LMCI, SNCI or MNCI), the performance of the cutting-plane algorithm depends on how quickly it can identify "good" cutsets. We propose to identify and maintain simultaneously multiple cutsets, called a *family*. Such a family of cutsets is obtained by partitioning the set of nodes N into L subsets $S_l, l = 1, ..., L$, such that $S_l \cap S_k = \emptyset$, for all $l \neq k$, and $\cup_{l=1,...,L} S_l = N$. Then, each subset $S_l, l = 1, ..., L$, induces two cutsets $(S_l, \bar{S}_l)$ and $(\bar{S}_l, S_l)$, and the corresponding partition of N determines a family of cutsets available for the generation of violated VI.

partition of N determines a family of cutsets available for the generation of violated VI.

An initial family of cutsets, for which we verify the violation of VI, is obtained by partitioning N into $|N|$ subsets, each containing a single node (i.e., $S_l = \{i\}, \forall\, i \in N, l = 1, ..., |N|$). To partition N into subsets of cardinalities greater than 1, we introduce a procedure called **GenerateMultiSet(M)**, which constructs iteratively multiple subsets S_l, each with a maximum cardinality of **M**. Since all types of VI (LCI, LMCI, SNCI and MNCI) have a better chance of being violated when the arcs in $(S_l, \bar{S}_l)$ show small fractional values $\bar{y}_{ij}$, the procedure attempts to construct the sets S_l with the objective of minimizing $\sum_{(i,j)\in(S_l,\bar{S}_l)} \bar{y}_{ij}$ and $\sum_{(j,i)\in(\bar{S}_l,S_l)} \bar{y}_{ji}$. At any step of the procedure, let S_l be a subset of N of cardinality smaller than **M**. Initially, the family contains one subset, S_1, having a single element (arbitrarily chosen). We denote by *free node* a node that is not included in any subset and by $\bar{N}$ the set of all free nodes. Also, for each free node j, let $w_j = \max\{\max_{i\in S_l} \bar{y}_{ij}, \max_{i\in S_l} \bar{y}_{ji}\}$. To achieve our objective, we identify the free node n such that

$$n = \text{argmax}_{j\in\bar{N}} \{w_j\}.$$

If n exists, then we add it to S_l and move to the next step: continue with the construction of S_l, if $|S_l| < \mathbf{M}$, or otherwise, proceed to the construction of S_{l+1} (by selecting arbitrarily some free node and then repeating the process). If, however, no free node is connected by an arc to at least one node in S_l, we choose n arbitrarily among the free nodes. The procedure stops when there are no more free nodes. The outline of the procedure is summarized next.

Procedure **GenerateMultiSet(M):**

1- Set $\bar{N} \leftarrow N, l \leftarrow 1$.

2- If $\bar{N} = \emptyset$, STOP.
Otherwise, select (arbitrarily) a node $m \in \bar{N}$.

3- Add m to S_l and update $\bar{N} \leftarrow \bar{N}\backslash\{m\}$.

4- If $|S_l| \geq \mathbf{M}$, $l \leftarrow l + 1$ and go to 2.

5- Find $n \in \bar{N}$ such that $n = \text{argmax}_{j\in\bar{N}} \{w_j\}$.

6- If n exists, $m \leftarrow n$ and go to 3.
Otherwise, go to 2.

Note that, the procedure attempts to include first in S_l a free node that is connected by an arc to at least one node in S_l to avoid generating VI that are aggregations of previously generated VI. Figure 3 illustrates such a situation, where we assume that all possible violated VI using cutsets induced by subsets of cardinality 1 have already been generated. Currently, the procedure is constructing subsets of cardinality 2. If we chose to include the free node 2 into the set $S_l = \{1\}$ to create the new set $S_l = \{1, 2\}$, we would identify a new cutset, but the resulting cutset inequality would just be the aggregation of the previously generated

cutset inequalities induced by {1} and {2}. Thus, this new cutset would not identify any violated VI.

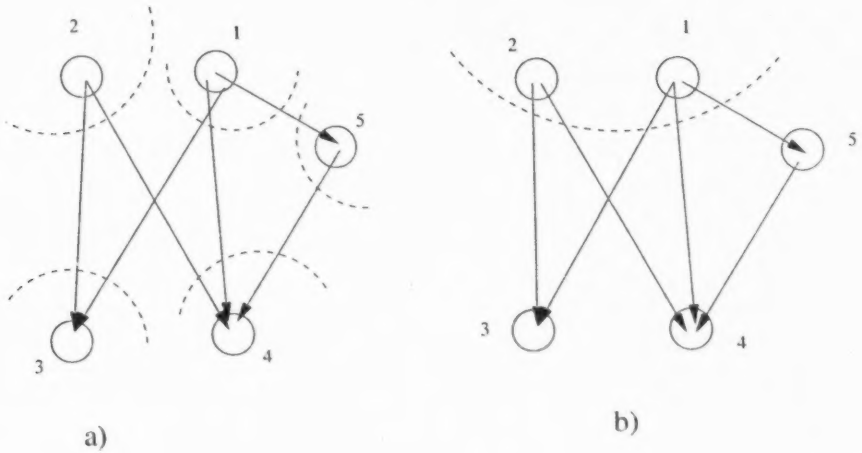

Figure 3: GenerateMultiSet example

Once a family of cutsets is determined by Procedure **GenerateMultiSet(M)**, all possible cuts are generated for this family. Then, new families of cutsets will be obtained by performing exchanges of nodes among subsets of the current family. The basic idea behind these exchanges is to obtain a new subset $S_{l'}$ from a subset S_l by moving a node n from some set S_k, $S_k \subset \bar{S}_l$, to S_l. These exchanges are performed by Procedure **MultiExchange**(W, W_N). The sets W and W_N contain, respectively, the indices l of all subsets S_l and the nodes $n \in N$ involved in some exchanges at previous calls to this procedure. These sets are used to ensure that the exchanges reach different subsets and involve different nodes, thus creating new cutsets at each iteration. The procedure considers at each step a set S_l and aims to identify and move to S_l the node n such that

$$n = \text{argmax}_{j \in (N \setminus W_N) \cap (\cup_{k \notin W, S_k \subset \bar{S}_l} S_k)} \{w_j\}.$$

Note that, $n \in N \setminus W_N$ is chosen among the set of nodes connected by an arc to at least one node in S_l. This strategy attempts to avoid generating VI that are aggregations of previously generated VI. Once n is identified, we move it from some set S_k to S_l. Then, the procedure repeats the process by considering subset S_k at the next iteration. The procedure starts with a set S_l not involved in previous exchanges (i.e., $l \notin W$). The procedure also stores in set V the indices of the subsets S_l considered at each iteration and stops whenever it finds a couple of subsets (S_l, S_k) involved in an exchange such that $k \in V$. This strategy identifies a cycle on which the nodes are moved around. This is illustrated in Figure 4, where the procedure stops after successively moving nodes from S_2 to S_1, from S_3 to S_2, from S_4 to S_3, and from S_2 to S_4. By doing so, subsets S_3 and S_4 have the same cardinality as before the exchanges, while S_1 and S_2 have, respectively, one additional element and one element less. This approach attempts to modify as less as possible the cardinality of the subsets. It is based on the observation that, to obtain a new family of cutsets, it is not necessary to

perform exchanges for each subset $S_l, l = 1, ..., L$. In fact, only a small number of exchanges allows to significantly modify the initial family. The outline of this procedure is presented next.

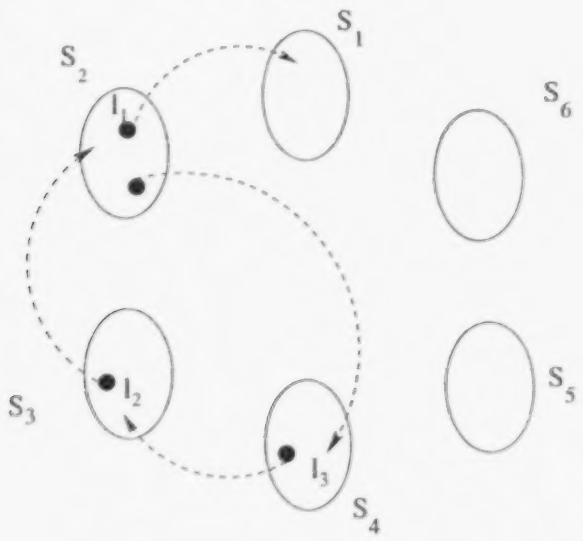

Figure 4: MultiExchange example

Procedure **MultiExchange(W, W_N):**

1- Start with a family of cutsets determined by the sets $(S_l)_{l=1,...,L}$; let $V \leftarrow \emptyset$; if $W = \{1, ..., L\}$, let $W \leftarrow \emptyset$; let $l \notin W$ corresponding to some set not involved in previous exchanges.

2- Find n such that

$$n = \text{argmax}_{j \in (N \setminus W_N) \cap (\cup_{k \notin W, S_k \subset \bar{S}_l} S_k)} \{w_j\}.$$

3- If $(\cup_{k \notin W, S_k \subset \bar{S}_l} S_k) = \emptyset$, let $W \leftarrow \emptyset$ and return to 2.

4- If n does not exist, let $W_N \leftarrow \emptyset$ and return to 2.

5- Let $S_k \subset \bar{S}_l$ such that $n \in S_k$; move n from S_k to S_l and let $W_N \leftarrow W_N \cup \{n\}$, $W \leftarrow W \cup \{l\}$ and $V \leftarrow V \cup \{l\}$.

6- If $k \in V$, STOP.
Otherwise, $l \leftarrow k$ and return to 2.

We show next how to use these two procedures within the cutting-plane algorithm.

6.2 Cutting-Plane Algorithm

The cutting-plane algorithm is divided into an initialization step and two improvement phases. The initialization step solves the LP relaxation of the problem. In the first phase, the cutting-plane procedure performs the generation of cuts based on the initial family of cutsets obtained by partitioning N into $|N|$ subsets, each containing a single element (i.e., $S_l = \{i\}, \forall i \in N, l = 1, ..., |N|$). The generation of cuts involves the addition of the corresponding VI, if any, to the formulation, and the resolution of the resulting LP relaxation. The first phase of the algorithm iterates on the initial family of cutsets until no further significant improvement is observed.

To obtain additional improvements, the second phase of the procedure is launched. In this phase, new families of cutsets are identified by performing procedure **GenerateMultiSet(M)**, where **M** varies from 2 to $\mathbf{M}_{max}$. For each family of cutsets, the algorithm performs the generation of cuts and identifies families of cutsets by applying procedure **MultiExchange**(W, W_N) for a given number of iterations, I_{max}. If the solution obtained at the end of this second phase improves upon that of the first phase, then the first phase is launched all over again. Thus, the proposed approach iterates over the first and second phases until no further improvement is obtained. In addition, to limit the computational effort, the second phase is invoked no more than T_{max} times.

The cutting-plane algorithm is summarized next. Here, Z and $(\bar{x}, \bar{y})$ represent the current LP optimal value and optimal solution, respectively, while Z_{last} is the LP optimal value at the last iteration during the first phase, and during the second phase, it is the LP optimal value at the end of the first phase. It is used to verify the stopping condition $Z - Z_{last} \leq \epsilon$, used in any of the two phases (ϵ is a parameter).

1. *Initialization step:*

   - $T \leftarrow 0$, $Z_{last} \leftarrow 0$.
   - Solve the LP relaxation; let Z be the optimal value and $(\bar{x}, \bar{y})$ be the corresponding optimal solution.
   - If $(\bar{x}, \bar{y})$ is feasible, go to 4.

2. *First phase:*

   - Initialize the family of cutsets by partitioning N into $|N|$ subsets (i.e., $S_l = \{ i\}, \forall i \in N, l = 1, ..., |N| = L$).
   - While $Z - Z_{last} > \epsilon$
     - Add cuts; if no cuts were found, go to 3.

- $Z_{last} \leftarrow Z$.
- Solve the LP relaxation; let Z be the optimal value and $(\bar{x}, \bar{y})$ be the corresponding optimal solution.
- If $(\bar{x}, \bar{y})$ is feasible, go to 4.

3. *Second phase:*

- If $T \geq T_{max}$, go to 4.
- $T \leftarrow T + 1$ and $Z_{last} \leftarrow Z$.
- For **M**= 2 to $\mathbf{M}_{max}$
  - Identify a family of cutsets by performing Procedure **GenerateMultiSet(M)**.
  - Add cuts.
  - If at least one cut has been added
    - Solve the LP relaxation; let Z be the optimal value and $(\bar{x}, \bar{y})$ be the corresponding optimal solution.
    - If $(\bar{x}, \bar{y})$ is feasible, go to 4.
  - $W \leftarrow \emptyset$ and $W_N \leftarrow \emptyset$.
  - For $I = 1$ to I_{max}
    - Identify a family of cutsets by performing Procedure **MultiExchange**(W, W_N).
    - Add cuts.
    - If at least one cut has been added
      - Solve the LP relaxation; let Z be the optimal value and $(\bar{x}, \bar{y})$ be the corresponding optimal solution.
      - If $(\bar{x}, \bar{y})$ is feasible, go to 4.
- If $Z - Z_{last} > \epsilon$, go to 2.

4. Return Z and $(\bar{x}, \bar{y})$.

The main objective of this study being to improve the quality of the lower bounds using a reasonable computational effort, it is interesting to evaluate the improvement that one might obtain by performing the overall algorithm compared to the first phase alone, and for how much additional computational effort. To this end, we identify two different approaches, the *simple cutting-plane algorithm (SCP)*, which is the given algorithm stopped at the end of the first phase, and the *advanced cutting-plane algorithm (ACP)*, which is the overall algorithm.

Adding the strong inequalities (SI) to the formulation of the MCND improves the quality of the lower bounds. Yet, for large-scale instances, solving the LP relaxation including all these inequalities is time consuming. Moreover, not all such inequalities are necessary. Hence, during the initialization step, the cutting-plane algorithm solves the weak LP relaxation, and subsequently, it adds only SI that are violated by the current LP solution. The outline of the *Add cuts* part in the algorithm becomes:

- Add violated SI.
- For each cutset $(S_l, \bar{S}_l), l = 1, ..., L$:
  - Add violated LCI.
  - Add violated LMCI.
  - Add violated SNCI or MNCI.

Using MNCI instead of SNCI can increase significantly the size of the problem since for each violated MNCI, we add one variable and the inequalities that define it. We might thus rightfully ask what is the improvement one can obtain by using MNCI instead of SNCI and for how much additional computational effort. To answer these issues, we treat these two families, SNCI and MNCI, separately in the analysis of the computational results.

To sum up, in our experiments, we identify and compare the performance of four different approaches:

- SCP-SNCI: the simple cutting-plane algorithm using SNCI;
- ACP-SNCI: the advanced cutting-plane algorithm using SNCI;
- SCP-MNCI: the simple cutting-plane algorithm using MNCI;
- ACP-MNCI: the advanced cutting-plane algorithm using MNCI.

7 Computational Results

To test the performance of the cutting-plane approaches, we report and analyze computational results on a large set of test problems. The implementations are performed in C++, using the option *dualopt* of CPLEX (version 7.1, 2002) to solve to optimality the LP relaxation of the MCND. All experiments are performed on a Sun Entreprise 10000 with 64 Gigabytes of RAM and operating under Solaris 2.7.

The values of the parameters in our implementation are $T_{max} = 5$, $I_{max} = 20$, $\mathbf{M}_{max} = \left\lceil \frac{N}{3} \right\rceil$ and $\epsilon = 0.1$. In addition, in Procedure **OpenCloseArcs**, we set $\epsilon_0 = 0.01$ and $\epsilon_1 = 0.7$.

These values have been selected following a preliminary computational study.

Our test sets for the experimental study are described in Crainic, Frangioni, and Gendron (2001) and Gendron and Crainic (1994, 1996). The same test sets are used in several papers on the MCND such as Crainic, Gendreau, and Farvolden (1998), Ghamlouche, Crainic, and Gendreau (2001, 2002), and Crainic, Gendron, and Hernu (2002). These problem tests consist of two series of general transshipment networks with one commodity per origin-destination and no parallel arcs. Positive transportation and fixed costs are associated with each arc. Note that, the transportation costs on each arc are the same for all commodities.

The first set of instances, denoted **C**, consists of 43 problem instances characterized by the number of nodes, the number of arcs, and the number of commodities denoted $|N|$, $|A|$, and $|K|$ respectively. Two additional letters are used to characterize the fixed cost level, "F" for high and "V" for low, relatively to the transportation cost, and the capacity level, "T" for tight and "L" for loose, compared to the total demand.

The second set of instances, denoted **R**, consists of nine sets of nine problem instances each. Each set is characterized by the same number of nodes, arcs, and commodities for the nine instances but with various levels of fixed cost and capacity ratios. Thus, $F = 0.01$ (F01), $F = 0.05$ (F05), and $F = 0.10$ (F10), are used to qualify the importance of the fixed costs compared to the transportation costs where the fixed cost ratio is computed as $F = |K| \sum_{(i,j)\in A} f_{ij} / \sum_{k\in K} d^k \sum_{(i,j)\in A} c^k_{ij}$. Similarly, $C = 1$ (C1), $C = 2$ (C2), and $C = 8$ (C8), are used to qualify the tightness of the total capacity compared to the total demand, where the capacity ratio is computed as $C = |A| \sum_{k\in K} d^k / \sum_{(i,j)\in A} u_{ij}$.

Our results, bounds, and computational times are compared to those obtained by computing the strong LP relaxation bounds using CPLEX (version 7.1, 2002), i.e., the strong inequalities are all added to the formulation (1)-(5) and the resulting model is provided to the solver. The gap relative to the optimum and the CPU time needed to achieve the lower bounds are our primary measures of performance. In order to assess the solution quality relative to the optimal solution, we tried to solve all instances using the branch-and-cut (B&C) of CPLEX (version 7.1, 2002) with default settings for all parameters (Chouman, Gendron, and Crainic 2001). A limit of 10 hours of computation was imposed for each instance. The results show that CPLEX is quite efficient at solving small size instances within the time allowed. However, when the problem size is increasing (especially the number of commodities), CPLEX is unable to find the optimal solution (not even a feasible solution for problems with 400 commodities) within the available computational time. For these instances, the best feasible solution found in Ghamlouche, Crainic, and Gendreau (2002) is used. It should be mentioned that these results also show that CPLEX was unable to generate any CI on a large set of instances.

To avoid overloading the paper, we report detailed results for only 13 selected problems in class **C**, with various dimensions (detailed results for all instances are presented in Appendix). These problems are difficult because they have high fixed costs (F) and tight capacities (T), except for P12. We decided to include P12, which has high fixed costs (F),

but loose capacities (L), since 10 hours of CPU time was not sufficient to solve its LP relaxation using CPLEX. Table 1 shows results for the selected problems. For each problem, we report the description (DESCR) by identifying the number of nodes, arcs, and commodities respectively, the optimal (indicated by *) or the best feasible solution (SOL), and the gap of the strong LP bound (GAP-LP) computed with respect to the best known solution SOL. The last four columns indicate the gap of the lower bound and the number of VI for each family generated in SCP-SNCI and ACP-SNCI, respectively. The CPU time elapsed in seconds is given under each solution. The letter (**t**) indicates that the solution is feasible but its optimality could not be verified within 10 hours with the B&C of CPLEX.

	DESCR			SCP-SNCI		ACP-SNCI	
	$\|N\|$, $\|A\|$, $\|K\|$	SOL	GAP-LP	GAP	(SI,LCI,LMCI,SNCI)	GAP	(SI,LCI,LMCI,SNCI)
P1	25, 100, 10	49899*	9.76	4.91	(68, 12, 12, 146)	3.22	(68, 14, 12, 240)
	FT	(40.6)	(0.3)	(0.7)		(44.2)	
P2	25, 100, 30	85530*	3.63	2.53	(107, 19, 20, 107)	2.21	(107, 22, 20, 137)
	FT	(534.2)	(1.6)	(1.3)		(40.1)	
P3	20, 230, 40	643036*	1.48	1.05	(100, 19, 18, 187)	1.04	(100, 19, 18, 188)
	FT	(671.7)	(0.6)	(1.3)		(71.3)	
P4	20, 300, 40	604198*	1.21	0.60	(92, 20, 22, 168)	0.56	(92, 20, 22, 176)
	FT	(59.8)	(0.6)	(1.3)		(70.9)	
P5	20, 230, 200	137271	4.17	4.17	(1252, 18, 17, 1211)	4.17	(1252, 18, 17, 1211)
	FT	(t)	(2431.1)	(272.4)		(726.5)	
P6	20, 300, 200	108252	4.26	4.14	(842, 13, 13, 882)	4.14	(842, 13, 13, 882)
	FT	(t)	(2605.1)	(302.2)		(765.0)	
P7	100, 400, 10	66364	23.02	15.17	(238, 22, 17, 582)	12.04	(325, 28, 17, 1061)
	FT	(t)	(13.0)	(21.7)		(857.9)	
P8	100, 400, 30	141278	15.44	11.14	(354, 47, 50, 717)	10.31	(386, 57, 50, 1090)
	FT	(t)	(110.2)	(62.3)		(1850.4)	
P9	30, 520, 100	98357	4.41	4.24	(652, 28, 42, 998)	4.24	(652, 28, 42, 998)
	FT	(t)	(863.1)	(143.4)		(910.7)	
P10	30, 700, 100	55082	2.58	2.45	(512, 31, 31, 787)	2.43	(519, 31, 31, 819)
	FT	(t)	(356.0)	(97.8)		(2687.1)	
P11	30, 520, 400	163352	8.32	8.29	(1585, 10, 6, 1579)	8.29	(1585, 10, 6, 1579)
	FT	(59820.1)	(11706.8)	(2674.2)		(5454.2)	
P12	30, 700, 400	144615	X	9.60	(2053, 14, 8, 1883)	9.60	(2053, 14, 8, 1883)
	FL	(57574.8)	(t)	(5367.9)		(9122.1)	
P13	30, 700, 400	138777	8.07	8.07	(1513, 7, 4, 1412)	8.07	(1513, 7, 4, 1412)
	FT	(48693.7)	(34057.0)	(3747.7)		(7104.1)	

Table 1: Detailed results for 13 selected instances

The results show that the bounds derived from SCP-SNCI and ACP-SNCI improve upon the strong LP bound. The average of gap improvement for problems P1 to P11 is 1.78% for SCP-SNCI and 2.33% for ACP-SNCI, while it can reach up to 7.85% and 11.13% (instance P7) for SCP-SNCI and ACP-SNCI, respectively. It should be pointed out that, even if this improvement could be seen as relatively modest, it is still quite significant: we have noted that a large number of nodes must be explored in the enumeration tree to get such improvement on some instances using the B&C of CPLEX. The improvement in the CPU time needed to compute the lower bounds is significant when compared to that needed by CPLEX to compute the

strong LP bound, especially when the problem dimension is increasing. For example, for P12, CPLEX does not manage to solve the strong LP relaxation in 10 hours while it requires about 90 minutes to compute the lower bound using SCP-SNCI and 152 minutes using ACP-SNCI.

			SCP-SNCI		ACP-SNCI		SCP-MNCI		ACP-MNCI	
$\|K\|$		GAP-LP	GAP	IMP	GAP	IMP	GAP	IMP	GAP	IMP
10	VL	2.03%	1.17%	0.86%	0.49%	1.54%	1.61%	0.42%	1.42%	0.61%
	FL	15.35%	10.51%	4.84%	7.40%	7.95%	11.49%	3.86%	8.58%	6.77%
	VT	16.39%	10.04%	6.35%	7.63%	8.76%	14.41%	1.98%	12.01%	4.38%
	FT	9.18%	6.74%	2.44%	6.05%	3.13%	7.33%	1.85%	6.65%	2.53%
30	VL	0.72%	0.47%	0.25%	0.31%	0.41%	0.57%	0.15%	0.56%	0.16%
	FL	9.53%	6.83%	2.70%	6.26%	3.27%	8.17%	1.36%	7.95%	1.58%
	VT	0.28%	0.26%	0.02%	0.26%	0.02%	0.26%	0.02%	0.26%	0.02%
	FT	1.84%	1.80%	0.04%	1.80%	0.04%	1.82%	0.02%	1.82%	0.02%
40	VL	0.74%	0.51%	0.23%	0.46%	0.28%	0.61%	0.13%	0.52%	0.22%
	FL	1.34%	0.82%	0.52%	0.80%	0.54%	1.11%	0.23%	1.11%	0.23%
	VT	1.17%	1.07%	0.10%	1.07%	0.10%	1.11%	0.06%	1.10%	0.07%
	FT	4.60%	4.53%	0.06%	4.53%	0.06%	4.53%	0.07%	4.53%	0.07%
100	VL	1.66%	1.38%	0.28%	1.37%	0.29%	1.52%	0.14%	1.52%	0.14%
	FL	3.49%	3.34%	0.15%	3.33%	0.16%	3.35%	0.14%	3.35%	0.14%
	VT	2.86%	2.83%	0.03%	2.83%	0.03%	2.83%	0.03%	2.83%	0.03%
	FT	6.13%	6.02%	0.11%	6.02%	0.11%	6.02%	0.11%	6.02%	0.11%
200	VL	1.80%	1.77%	0.03%	1.77%	0.03%	1.77%	0.03%	1.77%	0.03%
	FL	4.22%	4.15%	0.07%	4.15%	0.07%	4.15%	0.07%	4.15%	0.07%
	VT	3.71%	3.70%	0.01%	3.70%	0.01%	3.70%	0.01%	3.70%	0.01%
	FT	7.75%	7.71%	0.04%	7.71%	0.04%	7.71%	0.04%	7.71%	0.04%
400	VL	5.17%	5.17%	0.00%	5.17%	0.00%	5.17%	0.00%	5.17%	0.00%
	FL	8.20%	8.18%	0.02%	8.18%	0.02%	8.18%	0.02%	8.18%	0.02%
	VT									
	FT									

Table 2: Gap and improvement distribution according to problem characteristics, C problems

Table 2 summarizes the gap and the improvement distribution for all C problems according to the number of commodities and the different levels of fixed cost and capacity ratios. The gap and the improvement over the gap of the strong LP bound are given for the four approaches: SCP-SNCI, ACP-SNCI, SCP-MNCI, and ACP-MNCI. These results show that the bounds are improved over the strong LP bounds for the four approaches. Note also that the advanced cutting-plane algorithms (ACP-SNCI and ACP-MNCI) outperform the simple cutting-plane versions especially for small size problems. Moreover, the bounds derived from the simple and advanced cutting-plane algorithms using the SNCI outperform those using the MNCI. This is not really surprising since we have used a "rough" heuristic to solve the separation problem for the MNCI. Finally, it is clear that for all approaches the gap improvement decreases as the number of commodities increases. Also, for large-scale problems, with 200 and 400 commodities, the gap improvement is almost the same for all approaches.

Table 3 displays the CPU reduction (CPU) and the gap improvement (IMP) compared to CPLEX solving the strong LP relaxation, for large size problems. The average reduction in CPU time ranges from 35% up to 85% for the different approaches. Thus, the proposed approaches perform well and find better bounds in much shorter time (reduction of about 70% on average) when compared to the strong LP bounds computed by CPLEX.

	SCP-SNCI		ACP-SNCI		SCP-MNCI		ACP-MNCI	
$\|N\|, \|A\|, \|K\|$	CPU	IMP	CPU	IMP	CPU	IMP	CPU	IMP
20, 230, 200	80%	0.06%	45%	0.06%	76%	0.06%	51%	0.06%
20, 300, 200	85%	0.05%	58%	0.05%	82%	0.05%	57%	0.05%
30, 520, 400	71%	0.02%	35%	0.02%	67%	0.02%	49%	0.02%
30, 700, 400	84%	0.00%	62%	0.00%	81%	0.00%	76%	0.00%

Table 3: CPU reduction and gap improvement according to problem dimensions, large **C** problems

Our remarks and conclusions are confirmed by the results obtained on the **R** problems. Table 4 displays the aggregated results (9 problems in each category). Problems are grouped according to their difficulty from the easiest (F01,C1) to the most difficult (F10,C8). The results indicate that the bounds derived from the different cutting-plane approaches are better than the strong LP bounds. Moreover, the improvement increases with increasing problem difficulty. The gap improvement is slightly more significant for advanced cutting-plane versions than for simple cutting-plane versions. The average improvement is 0.48%, 0.53%, 0.28% and 0.30% for SCP-SNCI, ACP-SNCI, SCP-MNCI, and ACP-MNCI, respectively.

			SCP-SNCI		ACP-SNCI		SCP-MNCI		ACP-MNCI	
		GAP-LP	GAP	IMP	GAP	IMP	GAP	IMP	GAP	IMP
F01	C1	0.61%	0.56%	0.05%	0.56%	0.05%	0.56%	0.05%	0.56%	0.05%
	C2	1.19%	0.98%	0.21%	0.97%	0.22%	1.13%	0.06%	1.12%	0.07%
	C8	2.45%	1.76%	0.69%	1.68%	0.77%	2.18%	0.27%	2.16%	0.29%
F05	C1	4.28%	4.05%	0.23%	4.05%	0.23%	4.05%	0.23%	4.05%	0.23%
	C2	3.99%	3.63%	0.36%	3.61%	0.38%	3.71%	0.28%	3.66%	0.33%
	C8	4.33%	3.38%	0.95%	3.27%	1.06%	3.90%	0.43%	3.89%	0.44%
F10	C1	6.61%	6.24%	0.37%	6.24%	0.37%	6.27%	0.34%	6.27%	0.34%
	C2	6.10%	5.59%	0.51%	5.49%	0.61%	5.63%	0.47%	5.54%	0.56%
	C8	5.17%	4.18%	0.99%	4.11%	1.06%	4.75%	0.42%	4.75%	0.42%

Table 4: Gap distribution according to fixed cost and capacity level, **R** problems

Table 5 shows the CPU reduction and the gap improvement for the **R** problems according to problem dimensions. The results indicate that the gap improvement generally decreases as the problem size increases. However, the CPU reduction is more significant on larger instances. The results suggest that SCP-SNCI shows a good performance tradeoff when compared to the other versions (ACP-SNCI, SCP-MNCI and ACP-MNCI) since it provides a relatively high gap improvement in much shorter time.

		SCP-SNCI		ACP-SNCI		SCP-MNCI		ACP-MNCI	
$\|N\|, \|K\|,$	$\|A\|$	CPU	IMP	CPU	IMP	CPU	IMP	CPU	IMP
20, 100,	100	49%	0.50%	-61%	0.56%	46%	0.45%	-41%	0.53%
	200	60%	0.27%	-31%	0.29%	52%	0.08%	0%	0.08%
	300	39%	0.20%	-26%	0.23%	38%	0.12%	-22%	0.12%
20, 200,	100	41%	0.17%	-10%	0.18%	52%	0.15%	10%	0.15%
	200	74%	0.09%	23%	0.11%	72%	0.07%	44%	0.07%
	300	80%	0.16%	39%	0.16%	76%	0.08%	56%	0.08%

Table 5: CPU reduction and gap improvement according to problem dimensions, **R** problems

One might ask if the proposed cutset generation procedures, introduced in Section 6.1, are efficient to identify "good" cutsets of the network. To answer this question, we compare our cutting-plane results with those obtained by an explicit cutset enumeration procedure. It is clear that enumerating all possible cutsets is not feasible from a practical point of view. Thus, we enumerate all possible cutsets up to cardinality 4. The resulting cutset generation procedure is called ENUM4.

Table 6 reports the results for 33 selected problems: 6 in class **C** and 27 in class **R**. For each problem, we report the description of the problem, DESCR, by identifying the number of nodes, arcs, and commodities, as well as the level of difficulty. In addition to the gap of the strong LP bound (GAP-LP), the gap and the improvement over the strong LP bound of SCP-SNCI and ACP-SNCI, we report the gap with respect to the best known solution and the improvement over the strong LP bound of the solution obtained using enumeration (ENUM4). The results indicate that the improvement obtained using ENUM4 instead of the cutset generation heuristics (ACP-SNCI) is not significant for all problems. The maximal improvement over ACP-SNCI reaches 0.86% while the average improvement is 0.09% for all instances. Note that this improvement is obtained by allowing 10 to 15 times the CPU time needed in ACP-SNCI. Note also that for instances where ACP-SNCI does not manage to identify useful cutsets, ENUM4 also does not provide useful cutsets.

To evaluate more precisely the performance of the proposed cutting-plane algorithms, we have run the B&B and the B&C of CPLEX on the formulation obtained from ACP-SNCI. A limit of 10 hours of CPU time is imposed for each instance. The results are compared to those of the B&C of CPLEX on the strong formulation. We are interested in the category of problems for which CPLEX fails to find the optimal solution within the CPU time allowed. This category is composed of 68 problems (28 **C** problems and 40 **R** problems). The results show that the B&B on the new formulation found a better solution for 27 problems out of 68, compared to 19 better solutions found by the B&C on the new formulation, and 22 by the B&C on the strong formulation. Thus, within a limit of 10 hours of CPU time, the B&B of CPLEX performs better than its B&C when applied to the new formulation. For the 68 unsolved problems, Table 8 gives the distribution of the best feasible solutions found by the B&B on the new formulation when compared to those found by the B&C on the strong formulation. Among the 28 **C** problems, 3 are solved to optimality, a better solution

DESCR			SCP-SNCI		ACP-SNCI		ENUM4	
		GAP-LP	GAP	IMP	GAP	IMP	GAP	IMP
	VL	0.69%	0.55%	0.14%	0.50%	0.19%	0.00%	0.69%
25-100-10	FL	12.87%	8.60%	4.27%	8.11%	4.76%	7.25%	5.62%
	FT	9.76%	4.91%	4.85%	3.22%	6.54%	2.89%	6.87%
	VT	0.24%	0.16%	0.08%	0.13%	0.11%	0.13%	0.11%
25-100-30	FL	8.26%	5.73%	2.53%	5.46%	2.80%	5.42%	2.84%
	FT	3.63%	2.53%	1.10%	2.21%	1.42%	2.24%	1.39%
	F01-C1	0.58%	0.24%	0.34%	0.24%	0.34%	0.24%	0.34%
	F01-C2	3.84%	3.34%	0.50%	3.34%	0.50%	3.34%	0.50%
	F01-C8	4.27%	3.27%	1.00%	3.27%	1.00%	3.27%	1.00%
	F05-C1	2.32%	1.37%	1.95%	1.36%	1.96%	1.36%	1.96%
20-100-40	F05-C2	6.18%	5.14%	1.04%	5.13%	1.05%	4.99%	1.19%
	F05-C8	7.18%	5.98%	1.20%	5.53%	1.65%	5.52%	1.66%
	F10-C1	4.78%	2.34%	2.44%	2.01%	2.77%	1.62%	3.16%
	F10-C2	7.98%	3.65%	4.33%	3.14%	4.84%	2.92%	5.06%
	F10-C8	8.90%	3.77%	5.13%	3.24%	5.66%	2.73%	6.17%
	F01-C1	0.47%	0.46%	0.01%	0.46%	0.01%	0.46%	0.01%
	F01-C2	5.59%	5.24%	0.35%	5.24%	0.35%	5.24%	0.35%
	F01-C8	8.28%	7.78%	0.50%	7.78%	0.50%	7.78%	0.50%
	F05-C1	0.79%	0.77%	0.02%	0.77%	0.02%	0.77%	0.02%
20-300-100	F05-C2	4.24%	4.21%	0.03%	4.21%	0.03%	4.21%	0.03%
	F05-C8	7.46%	7.41%	0.05%	7.41%	0.05%	7.41%	0.05%
	F10-C1	2.94%	2.39%	0.55%	2.32%	0.62%	2.31%	0.63%
	F10-C2	4.13%	3.93%	0.20%	3.82%	0.31%	3.81%	0.32%
	F10-C8	7.92%	7.80%	0.12%	7.77%	0.15%	7.75%	0.17%
	F01-C1	1.63%	1.62%	0.01%	1.62%	0.01%	1.62%	0.01%
	F01-C2	3.38%	3.38%	0.00%	3.38%	0.00%	3.38%	0.00%
	F01-C8	10.07%	10.03%	0.04%	10.03%	0.04%	10.03%	0.04%
	F05-C1	2.02%	2.01%	0.01%	2.01%	0.01%	2.01%	0.01%
20-300-200	F05-C2	5.92%	5.69%	0.23%	5.69%	0.23%	5.69%	0.23%
	F05-C8	11.10%	10.90%	0.20%	10.90%	0.20%	10.90%	0.20%
	F10-C1	2.31%	2.20%	0.11%	2.20%	0.11%	2.18%	0.13%
	F10-C2	6.86%	6.61%	0.25%	6.61%	0.25%	6.60%	0.26%
	F10-C8	8.63%	8.04%	0.59%	8.04%	0.59%	7.99%	0.64%

Table 6: Gap distribution for 33 selected instances (comparison between cutset generation heuristics and partial enumeration)

PSet	$\|P\|$	OPT	<-2%	(-2% – -1%]	(-1%– 0%]	(0% – 1%]	(1% – %]	>2%
C	28	3	1	1	6	11	-	6
R	40	1	2	6	9	17	2	3

Table 7: Distribution of relative improvement for the 68 unsolved problems

is obtained in 17 cases and a worst solution in 8 cases. The improvement in the solution quality is greater than 2% for 6 problems. Among the 40 **R** problems, one problem is solved to optimality, 22 solutions are better and 17 solutions are worst. The gap for the worst solutions is more than 2% for only 2 problems.

These results emphasize the performance of the cutting-plane method to improve the quality of the bounds, and hence the formulation of the problem. This suggests implementing a B&C algorithm, where the cutting-plane procedure is used at each node in the enumeration tree.

8 Conclusion

In this paper, we have presented a cutting-plane algorithm based on cutset inequalities for the multicommodity capacitated fixed charge network design problem. We have described three families of valid inequalities: the cover inequalities, the minimum cardinality inequalities, and the network cutset inequalities. The latter constitutes a new family of VI derived from the cutset structure. We have developed computationally efficient separation heuristics to be used in conjunction with a cutset generation algorithm. We have also developed new lifting strategies, adapted to cover and minimum cardinality inequalities. Finally, we have presented computational results conducted on a large set of instances.

In the light of our computational study, we can conclude that the proposed cutting-plane algorithm achieves better lower bounds with competitive computational effort when compared to solving the strong LP relaxation with the state-of-the-art MIP software package CPLEX. This conclusion points out to interesting research avenues. The cutting-plane procedure can be included in a branch-and-cut algorithm, since the cuts used are valid at all nodes in the enumeration tree. An additional important advantage of our cutting-plane method is that it uses VI derived from any cutset structure, which constitutes the most common structure found in almost all network design problems (see Ortega and Wolsey 2003, for a recent example of the use of cutset structures in a branch-and-cut algorithm for the single commodity uncapacitated fixed charge network flow problem). Thus, it is interesting to investigate the usefulness of the proposed method to improve the formulations of other network design formulations.

References

[1] Aardal K. 1998, *Capacitated Facility Location: Separation Algorithms and Computational Experience,* Mathematical Programming 81, 149-175.

[2] Aardal K., Pochet Y. and Wolsey L.A. 1995, *Capacitated Facility Location: Valid Inequalities and Facets,* Mathematics of Operations Research 20, 562-582.

[3] Atamtürk A. 2002a, *On Splittable and Unsplittable Capacitated Network Design Arc-Set Polyhedra*, Mathematical Programming 92, 315-333.

[4] Atamtürk A. 2002a, *On Capacitated Network Design Cut-Set Polyhedra*, Mathematical Programming 92, 425-437.

[5] Balas E. 1975, *Facets of the Knapsack Polytope*, Mathematical Programming 8, 146-164.

[6] Barahona F. 1996, *Network Design Using Cut Inequalities*, SIAM Journal of Optimization 6, 823-837.

[7] Bienstock D., Chopra S., Günlük O. and Tsai C.Y. 1998, *Minimum Cost Capacity Installation for Multicommodity Network Flows*, Mathematical Programming 81, 177-199.

[8] Bienstock D. and Günlük O. 1996, *Capacitated Network Design-Polyhedral Structure and Computation,* INFORMS Journal on Computing 8, 243-259.

[9] Chouman M., Crainic T.G. and Gendron B. 2001, *Revue des inégalités valides pertinentes aux problèmes de conception de réseaux,* INFOR, forthcoming.

[10] CPLEX Optimization, Inc. 2002, *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*, version 7.1.

[11] Crainic, T.G. 1999, *Network Design in Freight Transportation*, European Journal of Operational Research, 122, (2), 272-288.

[12] Crainic T.G., Frangioni A. and Gendron B. 2002, *Bundle-Based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems,* Discrete Applied Mathematics, 112, 73-99.

[13] Crainic T.G. and Gendreau M. 2002, *Cooperative Parallel Tabu Search for Capacitated Network Design*, Journal of Heuristics 8, 601-627.

[14] Crainic T.G., Gendreau M. and Farvolden J.M. 2000, *A Simplex-Based Tabu Search Method for Capacitated Network Design*, INFORMS Journal on Computing 12, 223-236.

[15] Crainic T.G., Gendron B. and Hernu G. 2002 *A Slope Scaling/Lagrangean Perturbation Heuristic with Long-Term Memory for Multicommodity Capacitated Fixed-Charge Network Design*, Publication CRT, Centre de recherche sur les transports, Université de Montréal, Montréal.

[16] Hammer P.L., Johnson E.L. and Peled U.N. 1975, *Facets of Regular 0-1 Polytopes*, Mathematical Programming 8, 179-206.

[17] Gabrel V., Knippel A. and Minoux M. 1999, *Exact solution of multicommodity network optimization problems with general step cost functions*, Operations Research Letters 25, 15-23.

[18] Gendron B. and Crainic T.G. 1994, *Relaxations for Multicommodity Capacitated Network Design Problems*, Publication CRT-945, Centre de recherche sur les transports, Université de Montréal, Montréal.

[19] Gendron B. and Crainic T.G. 1996, *Bounding Procedures for Multicommodity Capacitated Fixed Charge Network Design Problems*, Publication CRT-96-06, Centre de recherche sur les transports, Université de Montréal, Montréal.

[20] Gendron B., Crainic T.G. and Frangioni A. 1998, *Multicommodity Capacitated Network Design, Telecommunications Network Planning*, Sansó B. and Soriano P. (eds.), Kluwer Academic Publishers, Norwell MA., 1-19.

[21] Ghamlouche I., Crainic T.G. et Gendreau M. 2001, *Cycle-Based Neighborhoods for Fixed-Charge Capacitated Multicommodity Network Design*, Operations Research, forthcoming.

[22] Ghamlouche I., Crainic T.G. et Gendreau M. 2002, *Path Relinking, Cycle-Based Neighborhoods and Capacitated Multicommodity Network Design*, Publication CRT, Centre de recherche sur les transports, Université de Montréal, Montréal.

[23] Gu Z., Nemhauser G.L. and Savelsbergh M.W.P. 1998, *Lifted Cover Inequalities for 0-1 Integer Programs: Computation*, INFORMS Journal on Computing 10, 427-437.

[24] Gu Z., Nemhauser G.L. and Savelsbergh M.W.P. 1999, *Lifted Cover Inequalities for 0-1 Integer Programs: Complexity*, INFORMS Journal on Computing 11, 117-123.

[25] Günlük O. 1999, *A Branch-and-Cut Algorithm for Capacitated Network Design Problems*, Mathematical Programming 86, 17-39.

[26] Holmberg K. and Yuan D. 2000, *A Lagrangean Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem*, Operations Research 48, 461-481.

[27] Leung J.M.Y. and Magnanti T.L. 1989, *Valid Inequalities and Facets of the Capacitated Plant Location Problems*, Mathematical Programming 44, 271-291.

[28] Magnanti T.L., Mirchandani P.B. and Vachani R. 1993, *The Convex Hull of Two Core Capacitated Network Design Problems*, Mathematical Programming 60, 233-250.

[29] Magnanti T.L., Mirchandani P.B. and Vachani R. 1995, *Modeling and Solving the Two-Facility Capacitated Network Loading Problem*, Operations Research 43, 142-157.

[30] Magnanti T.L. and Wong R.T. 1984, *Network Design and Transportation Planning: Models and Algorithms,* Transportation Science 18, 1-55.

[31] Martello S. and Toth P. 1990, *Knapsack Problems, Algorithms and Computer Implementations,* Wiley-Interscience.

[32] Martello S. and Toth P. 1997, *Upper Bounds and Algorithms for Hard 0-1 Knapsack Problems,* Operations Research 45, 768-778.

[33] Minoux M. 1989, *Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications,* Networks 19, 313-360.

[34] Nemhauser G.L., Savelsbergh M.W.P. and Sigismondi G. 1994, *MINTO, a Mixed Integer Optimizer,* Operations Research Letters 15, 47-58.

[35] Nemhauser G.L. and Wolsey L.A. 1988, *Integer and Combinatorial Optimization* , Wiley-Interscience.

[36] Ortega F. and Wolsey L. 2003, *A Branch-and-Cut Algorithm for the Single Commodity Uncapacitated Fixed Charge Network Flow Problem,* Networks 41, 143-158.

[37] Sellmann M., Kliewer G. and Koberstein A. 2002, *Capacitated Network Design, Cardinality Cuts and Coupled Variable Fixing Algorithms based on Lagrangian Relaxations,* Technical report tr-ri-02-234, Department of Computer Science, University of Paderborn.

[38] Van Roy T.J. and Wolsey L.A. 1987, *Solving Mixed Integer Programming Problems Using Automatic Reformulation,* Operations Research 35, 45-57.

[39] Wolsey L.A. 1975, *Faces of Linear Inequalities in 0-1 variables,* Mathematical Programming 8, 165-178.

[40] Wolsey L.A. 1998, *Integer Programming,* Wiley-Interscience.

[41] Xpress-MP 2002, *Xpress-MP Integrated into MIMI,* Modeling and Optimization Software, Dash Optimization Inc., New York, NY.

Appendix

Results for each problem instance is given in the following tables. Tables 9 to 19 show results for simple and advanced cutting-plane using the SNCI. While Tables 20 to 30 show results for simple and advanced cutting-plane using the MNCI. For each problem instance, the following information is displays:

- DESCR: The description of the problem given by the number of nodes, arcs, and commodities, respectively. Additional letters are used to characterize the fixed cost level relatively to the transportation cost and the capacity level compared to the total demand.

- SOL: The optimal (indicated by *) or the best feasible solution obtained by using the branch-and-cut (B&C) of CPLEX (version 7.1, 2002). A limit of 10 hours of computation was imposed for each instance. For the problems where CPLEX was unable to find feasible solution within the time allowed, we report the best feasible solution found in Ghamlouche, Crainic, and Gendreau (2002). The CPU time elapsed in seconds is given under each solution. The letter (**t**) indicates that the solution is feasible but its optimality could not be verified within 10 hours with the B&C of CPLEX.

- GAP-LP: Displays the gap of the strong LP bound computed with respect to the best solution (SOL).

- The last four columns indicate the gap of the lower bound and the number of valid inequalities for each family generated in the simple and advanced cutting-plane using SNCI or MNCI. The CPU time elapsed in second is given under each GAP.

All experiments are performed on a Sun Entreprise 10000 with 64 Gigabytes of RAM and operating under Solaris 2.7.

Descr			SCP-SNCI		ACP-SNCI	
$\|N\|,\|A\|,\|K\|$	Sol	gap-lp	gap	(si,lci,lmci,snci)	gap	(si,lci,lmci,snci)
25-100-10	14712*	0.69	0.55	(33, 13, 16, 52)	0.50	(33, 13, 16, 53)
V-L	(0.36)	(0.03)	(0.1)		(2.2)	
25-100-10	14941*	12.87	8.60	(137, 17, 30, 215)	8.11	(143, 20, 30, 241)
F-L	(53.6)	(0.5)	(1.3)		(32.6)	
25-100-10	49899*	9.76	4.91	(68, 12, 12, 146)	3.22	(68, 14, 12, 240)
F-T	(40.6)	(0.3)	(0.7)		(44.2)	
25-100-30	365272*	0.24	0.16	(75, 20, 20, 104)	0.13	(75, 20, 20, 116)
V-T	(16.6)	(0.6)	(0.6)		(13.1)	
25-100-30	37055*	8.26	5.73	(201, 28, 41, 249)	5.46	(204, 29, 41, 275)
F-L	(1727.9)	(4.7)	(4.7)		(72.3)	
25-100-30	85530*	3.63	2.53	(107, 19, 20, 107)	2.21	(107, 22, 20, 137)
F-T	(534.2)	(1.6)	(1.3)		(40.1)	
20-230-40	423848*	0.23	0.20	(114, 27, 47, 160)	0.19	(114, 28, 47, 160)
V-L	(10.4)	(0.3)	(0.8)		(34.8)	
20-230-40	371475*	0.71	0.43	(81, 13, 12, 132)	0.43	(81, 13, 12, 132)
V-T	(52.4)	(0.6)	(0.7)		(30.9)	
20-230-40	643036*	1.48	1.05	(100, 19, 18, 187)	1.04	(100, 19, 18, 188)
F-T	(671.7)	(0.6)	(1.3)		(71.3)	
20-300-40	429398*	0.33	0.33	(99, 27, 48, 166)	0.33	(99, 27, 48, 166)
V-L	(3.76)	(0.3)	(1.0)		(14.9)	
20-300-40	586077*	1.84	1.80	(150, 22, 30, 248)	1.80	(150, 22, 30, 248)
F-L	(145.5)	(1.0)	(2.2)		(49.3)	
20-300-40	464509*	0.77	0.60	(73, 13, 19, 135)	0.50	(73, 13, 19, 150)
V-T	(83.8)	(0.7)	(1.1)		(63.7)	
20-300-40	604198*	1.21	0.60	(92, 20, 22, 168)	0.56	(92, 20, 22, 176)
F-T	(59.8)	(0.6)	(1.3)		(70.9)	
20-230-200	94386	3.26	3.20	(1315, 17, 21, 1160)	3.20	(1315, 17, 21, 1160)
V-L	(t)	(1083.5)	(230.9)		(619.8)	
20-230-200	141737.4	5.84	6.66	(1532, 16, 23, 1239)	6.66	(1532, 16, 23, 1239)
F-L	(t)	(2235.1)	(327.3)		(744.5)	
20-230-200	97914	2.29	2.29	(1099, 20, 25, 1011)	2.29	(1099, 20, 25, 1014)
V-T	(t)	(594.3)	(187.0)		(598.1)	
20-230-200	137271	4.17	4.17	(1252, 18, 17, 1211)	4.17	(1252, 18, 17, 1211)
F-T	(t)	(2431.1)	(272.4)		(726.5)	
20-300-200	74972.4	2.46	2.46	(903, 10, 10, 920)	2.46	(903, 10, 10, 922)
V-L	(t)	(1332.1)	(250.7)		(811.9)	
20-300-200	117306	5.43	5.39	(1109, 12, 15, 1067)	5.39	(1109, 12, 15, 1067)
F-L	(t)	(3290.3)	(331.3)		(881.4)	
20-300-200	74991	1.31	1.26	(751, 11, 11, 781)	1.26	(751, 11, 11, 781)
V-T	(t)	(672.3)	(137.1)		(593.7)	
20-300-200	108252	4.26	4.14	(842, 13, 13, 882)	4.14	(842, 13, 13, 882)
F-T	(t)	(2605.1)	(302.2)		(765.0)	

Table 8: Computational Results Simple and Advanced Cutting-Plane-SNCI, C problems

Descr			SCP-SNCI		ACP-SNCI	
$\|N\|,\|A\|,\|K\|$	Sol	GAP-LP	GAP	(SI,LCI,LMCI,SNCI)	GAP	(SI,LCI,LMCI,SNCI)
100-400-10	28423*	3.37	1.79	(56, 15, 17, 119)	0.48	(56, 32, 17, 208)
V-L	(84.8)	(0.3)	(0.6)		(133.3)	
100-400-10	24436	17.82	12.43	(387, 18, 24, 555)	6.69	(535, 54, 24, 737)
F-L	(t)	(12.8)	(24.1)		(619.8)	
100-400-10	66364	23.02	15.17	(238, 22, 17, 582)	12.04	(325, 28, 17, 1061)
F-T	(t)	(13.0)	(21.7)		(857.9)	
100-400-30	385544	1.21	0.78	(148, 35, 39, 301)	0.49	(149, 35, 39, 481)
V-T	(t)	(6.1)	(2.9)		(694.4)	
100-400-30	50496	10.11	7.75	(895, 49, 74, 1162)	6.65	(1053, 61, 74, 1349)
F-L	(t)	(116.3)	(102.8)		(1200.2)	
100-400-30	141278	15.44	11.14	(354, 47, 50, 717)	10.31	(386, 57, 50, 1090)
F-T	(t)	(110.2)	(62.3)		(1850.4)	
30-520-100	53966	1.74	1.54	(467, 24, 26, 664)	1.54	(467, 25, 26, 673)
V-L	(t)	(65.9)	(40.5)		(1031.1)	
30-520-100	95294	5.37	5.25	(939, 39, 47, 1284)	5.25	(939, 39, 47, 1284)
F-L	(t)	(1106.5)	(240.8)		(900.2)	
30-520-100	52085	1.45	1.16	(350, 23, 31, 541)	1.16	(351, 23, 31, 543)
V-T	(t)	(59.9)	(25.5)		(885.1)	
30-520-100	98357	4.41	4.24	(652, 28, 42, 998)	4.24	(652, 28, 42, 998)
F-T	(t)	(863.1)	(143.4)		(910.7)	
30-700-100	47603*	0.61	0.61	(420, 26, 40, 590)	0.61	(420, 26, 40, 590)
V-L	(1736.1)	(57.5)	(36.8)		(469.4)	
30-700-100	60525	3.83	3.81	(768, 42, 57, 1061)	3.81	(768, 42, 57, 1061)
F-L	(t)	(758.0)	(160.5)		(1062.5)	
30-700-100	45944.5	1.88	1.61	(434, 23, 25, 725)	1.59	(437, 23, 25, 752)
V-T	(t)	(166.3)	(56.9)		(2771.9)	
30-700-100	55082	2.58	2.45	(512, 31, 31, 787)	2.43	(519, 31, 31, 819)
F-T	(t)	(356.0)	(97.8)		(2687.1)	
30-520-400	112997.5	1.09	1.08	(1416, 7, 7, 1235)	1.08	(1416, 7, 7, 1235)
V-L	(t)	(5356.8)	(1768.7)		(3745.6)	
30-520-400	159007	7.75	7.72	(1831, 16, 18, 1607)	7.72	(1836, 16, 18, 1629)
F-L	(33641.2)	(10649.4)	(3017.2)		(8009.6)	
30-520-400	119744	4.74	4.74	(1099, 5, 2, 1110)	4.74	(1099, 5, 2, 1110)
V-T	(20461.9)	(4910.5)	(1540.8)		(3233.7)	
30-520-400	163352	8.32	8.29	(1585, 10, 6, 1579)	8.29	(1585, 10, 6, 1579)
F-T	(59820.1)	(11706.8)	(2674.2)		(5454.2)	
30-700-400	103138	6.33	6.33	(1824, 16, 18, 1539)	6.33	(1834, 16, 18, 1552)
V-L	(17881.3)	(19428.7)	(3099.2)		(11329.2)	
30-700-400	144615	X	9.60	(2053, 14, 8, 1883)	9.60	(2053, 14, 8, 1883)
F-L	(57574.8)	(t)	(5367.9)		(9122.1)	
30-700-400	99606	5.61	5.61	(1344, 7, 7, 1267)	5.61	(1344, 7, 7, 1267)
V-T	(31405.7)	(10923.9)	(2367.1)		(5827.2)	
30-700-400	138777	8.07	8.07	(1513, 7, 4, 1412)	8.07	(1513, 7, 4, 1412)
F-T	(48693.7)	(34057.0)	(3747.7)		(7104.1)	

Table 9: Computational Results Simple and Advanced Cutting-Plane-SNCI, **C** problems

Descr 20,100,40	Sol	gap-lp	scp-snci gap	scp-snci (si,lci,lmci,snci)	acp-snci gap	acp-snci (si,lci,lmci,snci)
R10-F01-C1	200087*	0.58	0.24	(95, 24, 39, 112)	0.24	(95, 24, 39, 112)
	(10.7)	(0.2)	(0.5)		(4.7)	
R10-F05-C1	346813.5*	3.84	3.34	(215, 30, 48, 295)	3.34	(215, 30, 48, 295)
	(788.2)	(2.1)	(2.9)		(24.9)	
R10-F10-C1	488015*	4.27	3.27	(306, 30, 52, 424)	3.27	(306, 30, 52, 424)
	(563.3)	(4.9)	(4.9)		(28.4)	
R10-F01-C2	229196*	2.32	1.37	(98, 13, 16, 162)	1.36	(98, 13, 16, 174)
	(184.3)	(0.9)	(1.1)		(59.1)	
R10-F05-C2	411664*	6.18	5.14	(174, 25, 36, 266)	5.13	(174, 25, 36, 269)
	(5123.2)	(3.4)	(2.8)		(77.8)	
R10-F10-C2	609104*	7.18	5.98	(226, 27, 46, 346)	5.53	(232, 31, 46, 405)
	(27260.5)	(7.3)	(5.8)		(172.6)	
R10-F01-C8	486895*	4.78	2.34	(88, 12, 15, 157)	2.01	(88, 12, 15, 224)
	(391.3)	(5.8)	(3.8)		(110.4)	
R10-F05-C8	951056*	7.98	3.65	(115, 13, 22, 267)	3.14	(118, 14, 22, 398)
	(555.5)	(7.7)	(8.2)		(170.6)	
R10-F10-C8	1421740*	8.90	3.77	(139, 18, 24, 315)	3.24	(140, 19, 24, 460)
	(608.4)	(9.4)	(10.5)		(174.4)	

Table 10: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R10**

Descr 20,100,100	Sol	gap-lp	scp-snci gap	scp-snci (si,lci,lmci,snci)	acp-snci gap	acp-snci (si,lci,lmci,snci)
R11-F01-C1	714431*	0.51	0.44	(273, 17, 22, 309)	0.43	(273, 17, 22, 318)
	(392.5)	(8.0)	(5.5)		(80.0)	
R11-F05-C1	1265985	3.53	2.78	(560, 24, 42, 517)	2.78	(562, 24, 42, 522)
	(t)	(52.4)	(19.2)		(220.3)	
R11-F10-C1	1846295	5.50	4.81	(729, 28, 39, 577)	4.81	(729, 28, 39, 577)
	(t)	(130.5)	(32.8)		(113.1)	
R11-F01-C2	870451*	1.12	0.93	(229, 14, 17, 288)	0.92	(229, 14, 17, 292)
	(2960.1)	(19.8)	(7.1)		(100.5)	
R11-F05-C2	1623640*	3.42	2.46	(333, 22, 29, 314)	2.29	(341, 25, 29, 396)
	(21739.5)	(85.7)	(20.1)		(300.3)	
R11-F10-C2	2414060	4.68	3.31	(438, 23, 38, 351)	3.01	(442, 26, 38, 389)
	(t)	(104.9)	(26.0)		(241.8)	
R11-F01-C8	2294912*	0.26	0.10	(374, 25, 25, 50)	0.10	(374, 25, 25, 53)
	(31.5)	(15.8)	(13.3)		(29.4)	
R11-F05-C8	3507100*	0.56	0.26	(403, 26, 30, 45)	0.24	(403, 26, 30, 46)
	(79.8)	(27.6)	(21.2)		(46.3)	
R11-F10-C8	4579353*	0.27	0.23	(502, 27, 36, 40)	0.23	(502, 27, 36, 48)
	(48.5)	(21.8)	(18.3)		(32.2)	

Table 11: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R11**

DESCR 20,100,200	SOL	GAP-LP	SCP-SNCI GAP	SCP-SNCI (SI,LCI,LMCI,SNCI)	ACP-SNCI GAP	ACP-SNCI (SI,LCI,LMCI,SNCI)
R12-F01-C1	1639443*	0.83	0.82	(371, 11, 13, 280)	0.82	(372, 11, 13, 284)
	(4436.7)	(54.3)	(25.6)		(140.0)	
R12-F05-C1	3403074.2	3.99	3.86	(719, 21, 29, 416)	3.86	(720, 21, 29, 417)
	(t)	(400.8)	(79.2)		(412.3)	
R12-F10-C1	5276171	6.47	5.93	(856, 25, 40, 400)	5.91	(861, 25, 40, 414)
	(t)	(969.9)	(103.9)		(404.0)	
R12-F01-C2	2303557*	0.61	0.51	(346, 17, 18, 199)	0.49	(346, 17, 18, 225)
	(1739.8)	(122.9)	(38.9)		(177.0)	
R12-F05-C2	4669799*	0.82	0.51	(411, 20, 30, 168)	0.49	(411, 21, 30, 181)
	(3145.6)	(243.0)	(42.0)		(146.1)	
R12-F10-C2	7100019*	0.69	0.22	(518, 26, 42, 144)	0.22	(518, 26, 42, 152)
	(1683.8)	(624.2)	(52.4)		(167.8)	
R12-F01-C8	7635270*	0.02	0.02	(1174, 40, 38, 45)	0.02	(1174, 40, 38, 46)
	(78.1)	(68.1)	(64.6)		(73.6)	
R12-F05-C8	10067742*	0.01	0.01	(1372, 40, 36, 47)	0.01	(1372, 40, 36, 47)
	(61.5)	(54.0)	(94.0)		(97.9)	
R12-F10-C8	11967768*	0.00	0.00	(1507, 40, 40, 44)	0.00	(1507, 40, 40, 44)
	(50.3)	(48.9)	(62.2)		(64.0)	

Table 12: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R12**

DESCR 20,200,40	SOL	GAP-LP	SCP-SNCI GAP	SCP-SNCI (SI,LCI,LMCI,SNCI)	ACP-SNCI GAP	ACP-SNCI (SI,LCI,LMCI,SNCI)
R13-F01-C1	142947*	0.31	0.31	(172, 33, 60, 259)	0.31	(172, 33, 60, 259)
	(6.4)	(0.5)	(1.5)		(11.4)	
R13-F05-C1	263800*	4.82	4.69	(531, 32, 60, 677)	4.69	(531, 32, 60, 677)
	(547.9)	(7.4)	(12.5)		(85.6)	
R13-F10-C1	365800*	5.63	5.56	(697, 36, 68, 897)	5.56	(697, 36, 68, 897)
	(116.6)	(13.7)	(24.7)		(101.1)	
R13-F01-C2	150977*	1.76	1.19	(184, 24, 38, 294)	1.09	(184, 24, 38, 301)
	(90.1)	(0.7)	(2.4)		(90.5)	
R13-F05-C2	282682*	4.87	4.47	(373, 30, 49, 533)	4.47	(373, 30, 49, 533)
	(3551.8)	(11.1)	(9.8)		(84.9)	
R13-F10-C2	406790*	6.81	6.24	(457, 32, 55, 642)	6.18	(462, 33, 55, 656)
	(29695.6)	(21.2)	(16.8)		(168.2)	
R13-F01-C8	208088	3.53	2.32	(68, 12, 13, 175)	2.26	(68, 13, 13, 190)
	(t)	(7.2)	(4.2)		(230.1)	
R13-F05-C8	446997	5.47	4.03	(136, 27, 35, 279)	3.92	(138, 28, 35, 315)
	(t)	(31.3)	(11.0)		(174.4)	
R13-F10-C8	698280	5.31	4.07	(205, 31, 34, 361)	4.07	(205, 32, 34, 365)
	(t)	(54.6)	(17.1)		(201.9)	

Table 13: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R13**

Descr 20,200,100	Sol	gap-lp	SCP-SNCI gap	SCP-SNCI (si,lci,lmci,snci)	ACP-SNCI gap	ACP-SNCI (si,lci,lmci,snci)
R14-F01-C1	403414*	0.53	0.51	(453, 26, 43, 504)	0.51	(453, 26, 43, 504)
	(495.7)	(12.9)	(12.6)		(114.8)	
R14-F05-C1	750091	5.04	5.00	(998, 33, 57, 977)	5.00	(998, 33, 57, 977)
	(t)	(123.4)	(70.0)		(253.0)	
R14-F10-C1	1078595	7.30	7.23	(1286, 36, 60, 1147)	7.23	(1286, 36, 60, 1147)
	(t)	(319.2)	(163.7)		(380.9)	
R14-F01-C2	437607*	0.91	0.90	(328, 17, 22, 396)	0.90	(328, 17, 22, 396)
	(1327.3)	(18.3)	(11.6)		(94.0)	
R14-F05-C2	852330	4.35	4.29	(624, 26, 30, 718)	4.29	(624, 26, 30, 718)
	(t)	(223.1)	(56.7)		(259.3)	
R14-F10-C2	1230749.7	5.19	4.96	(781, 35, 38, 823)	4.96	(781, 35, 38, 823)
	(t)	(515.8)	(84.1)		(274.9)	
R14-F01-C8	670064	2.50	2.06	(145, 13, 13, 275)	1.88	(147, 13, 13, 343)
	(t)	(121.4)	(19.9)		(825.7)	
R14-F05-C8	1643376	6.09	5.53	(306, 26, 37, 390)	5.48	(311, 26, 37, 435)
	(t)	(321.1)	(70.0)		(691.8)	
R14-F10-C8	2630332	6.38	5.39	(372, 25, 50, 465)	5.37	(373, 25, 50, 483)
	(t)	(591.3)	(85.9)		(678.8)	

Table 14: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R14**

Descr 20,200,200	Sol	gap-lp	SCP-SNCI gap	SCP-SNCI (si,lci,lmci,snci)	ACP-SNCI gap	ACP-SNCI (si,lci,lmci,snci)
R15-F01-C1	1000787*	0.66	0.65	(716, 17, 16, 684)	0.65	(716, 17, 16, 684)
	(5384.7)	(95.5)	(75.1)		(355.8)	
R15-F05-C1	1979413	4.44	4.23	(1559, 30, 29, 1245)	4.23	(1559, 30, 29, 1245)
	(t)	(1923.4)	(349.1)		(788.9)	
R15-F10-C1	2949264	7.47	7.17	(2017, 33, 40, 1444)	7.17	(2017, 33, 40, 1444)
	(t)	(4390.3)	(690.0)		(1124.0)	
R15-F01-C2	1148604	0.91	0.90	(551, 12, 9, 598)	0.90	(551, 12, 9, 601)
	(t)	(207.7)	(78.8)		(581.1)	
R15-F05-C2	2488753	3.75	3.75	(974, 28, 30, 911)	3.75	(974, 28, 30, 911)
	(t)	(2478.0)	(299.8)		(678.7)	
R15-F10-C2	3972667	7.36	7.33	(1288, 29, 37, 963)	7.27	(1288, 29, 37, 963)
	(t)	(9459.6)	(422.9)		(803.3)	
R15-F01-C8	2301326.6	1.10	0.83	(275, 17, 23, 197)	0.81	(276, 17, 23, 290)
	(t)	(525.9)	(115.7)		(1135.5)	
R15-F05-C8	5573412.8*	0.50	0.47	(424, 24, 44, 149)	0.43	(425, 24, 44, 210)
	(16107.9)	(967.7)	(189.6)		(737.8)	
R15-F10-C8	8696932*	0.21	0.21	(562, 28, 51, 58)	0.21	(562, 28, 51, 58)
	(18930.9)	(1224.4)	(267.4)		(295.8)	

Table 15: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R15**

DESCR 20,300,40	SOL	GAP-LP	SCP-SNCI GAP	SCP-SNCI (SI,LCI,LMCI,SNCI)	ACP-SNCI GAP	ACP-SNCI (SI,LCI,LMCI,SNCI)
R16-F01-C1	136161*	0.00	0.00	(253, 35, 65, 394)	0.00	(253, 35, 65, 394)
	(1.1)	(0.5)	(2.9)		(8.1)	
R16-F05-C1	239500*	3.95	3.93	(790, 38, 70, 1178)	3.93	(790, 38, 70, 1178)
	(1641.3)	(12.8)	(32.4)		(160.5)	
R16-F10-C1	325671*	4.50	4.46	(1042, 36, 69, 1471)	4.46	(1042, 36, 69, 1471)
	(4345.9)	(22.5)	(65.0)		(205.4)	
R16-F01-C2	138532*	0.32	0.32	(230, 31, 51, 372)	0.32	(230, 31, 51, 391)
	(27.6)	(1.1)	(3.6)		(52.5)	
R16-F05-C2	241801*	2.44	2.19	(635, 39, 61, 977)	2.19	(635, 39, 61, 977)
	(851.9)	(16.6)	(30.3)		(165.6)	
R16-F10-C2	337762*	4.47	3.96	(761, 38, 67, 1117)	3.96	(761, 38, 67, 1117)
	(8173.5)	(37.4)	(45.8)		(176.2)	
R16-F01-C8	169502	4.67	3.59	(127, 16, 21, 266)	3.52	(128, 17, 21, 289)
	(t)	(3.9)	(5.0)		(372.1)	
R16-F05-C8	352976	7.37	5.93	(263, 28, 39, 504)	5.83	(271, 29, 39, 536)
	(t)	(63.3)	(21.1)		(497.5)	
R16-F10-C8	541626	8.92	8.12	(328, 29, 45, 587)	8.06	(336, 30, 45, 627)
	(t)	(115.8)	(38.6)		(531.9)	

Table 16: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R16**

DESCR 20,300,100	SOL	GAP-LP	SCP-SNCI GAP	SCP-SNCI (SI,LCI,LMCI,SNCI)	ACP-SNCI GAP	ACP-SNCI (SI,LCI,LMCI,SNCI)
R17-F01-C1	354138*	0.47	0.46	(680, 36, 65, 698)	0.46	(680, 36, 65, 698)
	(332.1)	(17.7)	(29.6)		(219.7)	
R17-F05-C1	651025	5.59	5.24	(1554, 41, 74, 1626)	5.24	(1554, 41, 74, 1626)
	(t)	(182.5)	(215.5)		(556.0)	
R17-F10-C1	917956	8.28	7.78	(1942, 42, 78, 1881)	7.78	(1947, 42, 78, 1882)
	(t)	(432.9)	(340.5)		(1034.3)	
R17-F01-C2	370590*	0.79	0.77	(586, 25, 36, 749)	0.77	(586, 25, 36, 749)
	(2258.8)	(38.7)	(36.4)		(233.9)	
R17-F05-C2	708698	4.24	4.21	(1071, 39, 51, 1262)	4.21	(1071, 39, 51, 1262)
	(t)	(331.0)	(155.8)		(319.6)	
R17-F10-C2	1029242	7.46	7.41	(1217, 35, 52, 1357)	7.41	(1217, 35, 52, 1357)
	(t)	(841.1)	(212.1)		(547.2)	
R17-F01-C8	505310	2.94	2.39	(249, 12, 10, 437)	2.32	(249, 12, 10, 479)
	(t)	(141.4)	(37.0)		(940.5)	
R17-F05-C8	1112076	4.13	3.93	(405, 24, 31, 569)	3.82	(408, 24, 31, 603)
	(t)	(634.5)	(106.2)		(1050.4)	
R17-F10-C8	1847334	7.92	7.80	(502, 28, 34, 598)	7.77	(506, 28, 34, 612)
	(t)	(1551.1)	(141.9)		(1035.7)	

Table 17: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R17**

DESCR 20,300,200	SOL	GAP-LP	SCP-SNCI GAP	SCP-SNCI (SI,LCI,LMCI,SNCI)	ACP-SNCI GAP	ACP-SNCI (SI,LCI,LMCI,SNCI)
R18-F01-C1	828559	1.63	1.62	(1205, 22, 22, 1101)	1.62	(1205, 22, 22, 1101)
	(t)	(241.6)	(170.6)		(779.8)	
R18-F05-C1	1542197	3.38	3.38	(2522, 37, 50, 2145)	3.38	(2522, 37, 50, 2145)
	(t)	(7144.3)	(949.5)		(1572.5)	
R18-F10-C1	2292684	10.07	10.03	(2828, 41, 55, 2253)	10.03	(2828, 41, 55, 2253)
	(t)	(17349.7)	(1285.4)		(1916.2)	
R18-F01-C2	921762	2.02	2.01	(897, 12, 11, 963)	2.01	(897, 12, 11, 963)
	(t)	(423.7)	(178.8)		(795.9)	
R18-F05-C2	1866215	5.92	5.69	(1438, 17, 17, 1499)	5.69	(1438, 17, 17, 1499)
	(t)	(9144.4)	(555.9)		(1193.8)	
R18-F10-C2	2895982	11.10	10.90	(1564, 22, 18, 1408)	10.90	(1564, 22, 18, 1408)
	(t)	(13964.3)	(801.3)		(1402.4)	
R18-F01-C8	1485690.8	2.31	2.20	(366, 10, 10, 562)	2.20	(369, 10, 10, 572)
	(t)	(1408.0)	(209.1)		(1459.4)	
R18-F05-C8	4010736.7	6.86	6.61	(517, 22, 34, 522)	6.61	(517, 22, 34, 529)
	(t)	(4536.6)	(493.8)		(3092.5)	
R18-F10-C8	6663506.4	8.63	8.04	(617, 30, 47, 563)	8.04	(617, 30, 47, 563)
	(t)	(9860.1)	(664.8)		(1358.8)	

Table 18: Computational Results Simple and Advanced Cutting-Plane-SNCI, **R18**

Descr $\lvert N\rvert,\lvert A\rvert,\lvert K\rvert$	Sol	gap-lp	scp-mnci gap	scp-mnci (si,lci,lmci,mnci)	acp-mnci gap	acp-mnci (si,lci,lmci,mnci)
25-100-10	14712*	0.69	0.63	(33, 13, 16, 29)	0.61	(33, 14, 16, 29)
V-L	(0.36)	(0.03)	(0.1)		(9.1)	
25-100-10	14941*	12.87	9.74	(108, 17, 30, 140)	9.35	(116, 19, 30, 377)
F-L	(53.6)	(0.5)	(1.3)		(32.5)	
25-100-10	49899*	9.76	9.06	(68, 12, 12, 131)	6.84	(69, 19, 12, 144)
F-T	(40.6)	(0.3)	(1.0)		(20.8)	
25-100-30	365272*	0.24	0.21	(75, 20, 20, 52)	0.21	(75, 20, 20, 52)
V-T	(16.6)	(0.6)	(0.6)		(12.0)	
25-100-30	37055*	8.26	6.75	(178, 28, 41, 108)	6.53	(188, 30, 41, 133)
F-L	(1727.9)	(4.7)	(3.2)		(56.6)	
25-100-30	85530*	3.63	3.43	(99, 19, 20, 48)	3.43	(99, 19, 20, 48)
F-T	(534.2)	(1.6)	(1.2)		(14.3)	
20-230-40	423848*	0.23	0.20	(114, 27, 47, 80)	0.19	(114, 28, 47, 80)
V-L	(10.4)	(0.3)	(1.2)		(83.1)	
20-230-40	371475*	0.71	0.60	(84, 13, 12, 61)	0.43	(84, 13, 12, 128)
V-T	(52.4)	(0.6)	(2.1)		(52.9)	
20-230-40	643036*	1.48	1.10	(108, 19, 18, 125)	1.10	(108, 19, 18, 128)
F-T	(671.7)	(0.6)	(2.8)		(63.5)	
20-300-40	429398*	0.33	0.33	(99, 27, 48, 78)	0.33	(99, 27, 48, 78)
V-L	(3.76)	(0.3)	(1.3)		(43.9)	
20-300-40	586077*	1.84	1.82	(147, 22, 30, 103)	1.82	(147, 22, 30, 103)
F-L	(145.5)	(1.0)	(2.9)		(71.4)	
20-300-40	464509*	0.77	0.63	(74, 13, 19, 57)	0.62	(74, 16, 19, 97)
V-T	(83.8)	(0.7)	(2.3)		(201.5)	
20-300-40	604198*	1.21	1.12	(100, 19, 21, 50)	1.12	(100, 19, 21, 51)
F-T	(59.8)	(0.6)	(2.2)		(56.2)	
20-230-200	94386	3.26	3.20	(1326, 18, 21, 349)	3.20	(1326, 18, 21, 349)
V-L	(t)	(1083.5)	(290.2)		(566.3)	
20-230-200	141737.4	6.84	6.66	(1510, 16, 23, 382)	6.66	(1510, 16, 23, 382)
F-L	(t)	(2235.1)	(383.1)		(681.7)	
20-230-200	97914	2.29	2.29	(1104, 20, 25, 306)	2.29	(1104, 20, 25, 306)
V-T	(t)	(594.3)	(228.0)		(506.8)	
20-230-200	137271	4.17	4.17	(1263, 18, 17, 374)	4.17	(1263, 18, 17, 374)
F-T	(t)	(2431.1)	(383.1)		(660.4)	
20-300-200	74972.4	2.46	2.46	(942, 10, 10, 288)	2.46	(942, 10, 10, 288)
V-L	(t)	(1332.1)	(281.0)		(643.2)	
20-300-200	117306	5.43	5.39	(1116, 12, 15, 291)	5.39	(1116, 12, 15, 291)
F-L	(t)	(3290.3)	(458.3)		(847.4)	
20-300-200	74991	1.31	1.26	(741, 11, 11, 243)	1.26	(741, 11, 11, 243)
V-T	(t)	(672.3)	(170.9)		(519.0)	
20-300-200	108252	4.26	4.14	(839, 13, 13, 254)	4.14	(839, 13, 13, 254)
F-T	(t)	(2605.1)	(319.0)		(643.4)	

Table 19: Computational Results Simple and Advanced Cutting-Plane-MNCI, C problems

Descr $\|N\|,\|A\|,\|K\|$	Sol	gap-lp	scp-mnci gap	scp-mnci (si,lci,lmci,mnci)	acp-mnci gap	acp-mnci (si,lci,lmci,mnci)
100-400-10	28423*	3.37	2.59	(52, 15, 17, 96)	2.24	(58, 99, 17, 1895)
V-L	(84.8)	(0.3)	(2.2)		(1349.5)	
100-400-10	24436	17.82	13.25	(399, 17, 24, 502)	7.82	(528, 72, 24, 6795)
F-L	(t)	(12.8)	(45.2)		(43402.1)	
100-400-10	66364	23.02	19.77	(239, 21, 17, 817)	17.19	(276, 40, 17, 6273)
F-T	(t)	(13.0)	(47.3)		(6154.3)	
100-400-30	385544	1.21	0.94	(147, 35, 39, 301)	0.92	(147, 38, 39, 601)
V-T	(t)	(6.1)	(5.9)		(270.6)	
100-400-30	50496	10.11	7.92	(930, 49, 74, 861)	6.78	(1051, 61, 74, 2126)
F-L	(t)	(116.3)	(213.6)		(1914.6)	
100-400-30	141278	15.44	12.91	(304, 48, 50, 684)	12.47	(338, 59, 51, 2598)
F-T	(t)	(110.2)	(62.9)		(1332.1)	
30-520-100	53966	1.74	1.61	(466, 24, 26, 232)	1.59	(471, 26, 26, 243)
V-L	(t)	(65.9)	(46.5)		(712.7)	
30-520-100	95294	5.37	5.25	(932, 40, 47, 495)	5.25	(932, 40, 47, 495)
F-L	(t)	(1106.5)	(365.6)		(739.4)	
30-520-100	52085	1.45	1.33	(341, 23, 31, 219)	1.33	(341, 23, 31, 219)
V-T	(t)	(59.9)	(32.1)		(351.7)	
30-520-100	98357	4.41	4.24	(661, 27, 40, 430)	4.24	(661, 27, 40, 432)
F-T	(t)	(863.1)	(253.5)		(625.8)	
30-700-100	47603*	0.61	0.61	(436, 27, 41, 251)	0.61	(436, 27, 41, 251)
V-L	(1736.1)	(57.5)	(57.1)		(467.5)	
30-700-100	60525	3.83	3.81	(784, 43, 59, 451)	3.81	(784, 43, 59, 451)
F-L	(t)	(758.0)	(247.8)		(782.2)	
30-700-100	45944.5	1.88	1.71	(422, 23, 25, 260)	1.71	(422, 23, 25, 260)
V-T	(t)	(166.3)	(76.7)		(603.8)	
30-700-100	55082	2.58	2.47	(507, 31, 32, 353)	2.47	(507, 31, 32, 353)
F-T	(t)	(?56.0)	(146.5)		(626.4)	
30-520-400	112997.5	1.09	1.08	(1422, 7, 7, 349)	1.08	(1422, 7, 7, 349)
V-L	(t)	(5356.8)	(1949.1)		(3223.7)	
30-520-400	159007	7.75	7.72	(1821, 16, 18, 430)	7.72	(1821, 16, 18, 430)
F-L	(33641.2)	(10649.4)	(3385.3)		(4632.7)	
30-520-400	119744	4.74	4.74	(1100, 5, 2, 306)	4.74	(1100, 5, 2, 306)
V-T	(20461.9)	(4910.5)	(1776.1)		(3027.7)	
30-520-400	163352	8.32	8.29	(1618, 11, 6, 413)	8.29	(1618, 11, 6, 413)
F-T	(59820.1)	(11706.8)	(3323.4)		(4598.5)	
30-700-400	103138	6.33	6.33	(1870, 16, 18, 445)	6.33	(1870, 16, 18, 445)
V-L	(17881.3)	(19428.7)	(3492.2)		(5336.6)	
30-700-400	144615	X	9.60	(2039, 14, 8, 487)	9.60	(2039, 14, 8, 487)
F-L	(57574.8)	(t)	(6101.5)		(7994.2)	
30-700-400	99606	5.61	5.61	(1349, 7, 7, 352)	5.61	(1349, 7, 7, 352)
V-T	(31405.7)	(10923.9)	(2707.8)		(4413.6)	
30-700-400	138777	8.07	8.07	(1529, 7, 4, 376)	8.07	(1529, 7, 4, 376)
F-T	(48693.7)	(34057.0)	(5019.6)		(6849.0)	

Table 20: Computational Results Simple and Advanced Cutting-Plane-MNCI, C problems

DESCR 20,100,40	SOL	GAP-LP	SCP-MNCI GAP	SCP-MNCI (SI,LCI,LMCI,MNCI)	ACP-MNCI GAP	ACP-MNCI (SI,LCI,LMCI,MNCI)
R10-F01-C1	200087*	0.58	0.24	(95, 24, 39, 43)	0.24	(95, 24, 39, 43)
	(10.7)	(0.2)	(0.6)		(16.8)	
R10-F05-C1	346813.5*	3.84	3.34	(215, 30, 48, 149)	3.34	(215, 30, 48, 149)
	(788.2)	(2.1)	(3.9)		(28.8)	
R10-F10-C1	488015*	4.27	3.28	(298, 30, 52, 204)	3.28	(298, 30, 52, 204)
	(563.3)	(4.9)	(6.4)		(31.5)	
R10-F01-C2	229196*	2.32	2.00	(94, 13, 16, 63)	1.92	(94, 14, 16, 74)
	(184.3)	(0.9)	(1.1)		(43.3)	
R10-F05-C2	411664*	6.18	5.45	(165, 25, 36, 117)	5.45	(165, 25, 36, 120)
	(5123.2)	(3.4)	(3.7)		(27.5)	
R10-F10-C2	609104*	7.18	6.05	(234, 28, 46, 235)	5.59	(243, 31, 46, 352)
	(27260.5)	(7.3)	(9.3)		(80.8)	
R10-F01-C8	486895*	4.78	3.44	(87, 13, 16, 74)	3.27	(87, 14, 16, 78)
	(391.3)	(5.8)	(2.8)		(42.5)	
R10-F05-C8	951056*	7.98	6.05	(120, 13, 22, 75)	6.05	(120, 13, 22, 75)
	(555.5)	(7.7)	(3.8)		(27.2)	
R10-F10-C8	1421740*	8.90	6.61	(134, 18, 24, 56)	6.61	(134, 18, 24, 56)
	(608.4)	(9.4)	(3.6)		(23.9)	

Table 21: Computational Results Simple and Advanced Cutting-Plane-MNCI, **R10**

DESCR 20,100,100	SOL	GAP-LP	SCP-MNCI GAP	SCP-MNCI (SI,LCI,LMCI,MNCI)	ACP-MNCI GAP	ACP-MNCI (SI,LCI,LMCI,MNCI)
R11-F01-C1	714431*	0.51	0.45	(275, 17, 22, 113)	0.45	(275, 17, 22, 113)
	(392.5)	(8.0)	(6.6)		(54.3)	
R11-F05-C1	1265985	3.53	2.78	(551, 24, 42, 203)	2.78	(551, 24, 42, 203)
	(t)	(52.4)	(21.9)		(70.5)	
R11-F10-C1	1846295	5.50	4.81	(729, 28, 39, 229)	4.81	(729, 28, 39, 229)
	(t)	(130.5)	(38.8)		(103.8)	
R11-F01-C2	870451*	1.12	1.10	(218, 14, 17, 122)	1.10	(218, 14, 17, 122)
	(2960.1)	(19.8)	(8.1)		(59.9)	
R11-F05-C2	1623640*	3.42	2.57	(343, 22, 29, 145)	2.15	(347, 26, 29, 158)
	(21739.5)	(85.7)	(21.6)		(137.0)	
R11-F10-C2	2414060	4.68	3.38	(430, 23, 38, 150)	3.03	(446, 27, 38, 173)
	(t)	(104.9)	(28.1)		(160.2)	
R11-F01-C8	2294912*	0.26	0.21	(374, 25, 25, 26)	0.21	(374, 25, 25, 26)
	(31.5)	(15.8)	(17.5)		(37.1)	
R11-F05-C8	3507100*	0.56	0.27	(404, 26, 30, 26)	0.27	(404, 26, 30, 26)
	(79.8)	(27.6)	(16.0)		(39.0)	
R11-F10-C8	4579353*	0.27	0.23	(502, 27, 36, 12)	0.23	(502, 27, 36, 12)
	(48.5)	(21.8)	(17.3)		(47.8)	

Table 22: Computational Results Simple and Advanced Cutting-Plane-MNCI, **R11**

DESCR 20,100,200	SOL	GAP-LP	SCP-MNCI GAP	SCP-MNCI (SI,LCI,LMCI,MNCI)	ACP-MNCI GAP	ACP-MNCI (SI,LCI,LMCI,MNCI)
R12-F01-C1	1639443*	0.83	0.82	(371, 11, 13, 89)	0.82	(371, 11, 13, 89)
	(4436.7)	(54.3)	(24.1)		(102.5)	
R12-F05-C1	3403074.2	3.99	3.86	(719, 21, 29, 164)	3.86	(720, 21, 29, 169)
	(t)	(400.8)	(80.5)		(339.9)	
R12-F10-C1	5276171	6.47	6.06	(847, 25, 40, 145)	6.06	(847, 25, 40, 145)
	(t)	(969.9)	(96.2)		(209.8)	
R12-F01-C2	2303557*	0.61	0.53	(329, 17, 18, 73)	0.53	(329, 17, 18, 73)
	(1739.8)	(122.9)	(36.7)		(127.3)	
R12-F05-C2	4669799*	0.82	0.55	(402, 20, 30, 58)	0.53	(404, 22, 30, 62)
	(3145.6)	(243.0)	(46.0)		(196.8)	
R12-F10-C2	7100019*	0.69	0.22	(522, 26, 42, 76)	0.22	(522, 26, 42, 76)
	(1683.8)	(624.2)	(55.5)		(177.2)	
R12-F01-C8	7635270*	0.02	0.02	(1174, 40, 38, 0)	0.02	(1174, 40, 38, 0)
	(78.1)	(68.1)	(46.9)		(81.7)	
R12-F05-C8	10067742*	0.01	0.01	(1372, 40, 36, 0)	0.01	(1372, 40, 36, 0)
	(61.5)	(54.0)	(51.3)		(108.7)	
R12-F10-C8	11967768*	0.00	0.00	(1507, 40, 40, 0)	0.00	(1507, 40, 40, 0)
	(50.3)	(48.9)	(46.2)		(98.1)	

Table 23: Computational Results Simple and Advanced Cutting-Plane-MNCI, **R12**

DESCR 20,200,40	SOL	GAP-LP	SCP-MNCI GAP	SCP-MNCI (SI,LCI,LMCI,MNCI)	ACP-MNCI GAP	ACP-MNCI (SI,LCI,LMCI,MNCI)
R13-F01-C1	142947*	0.31	0.31	(176, 33, 60, 136)	0.31	(176, 33, 60, 136)
	(6.4)	(0.5)	(2.5)		(31.3)	
R13-F05-C1	263800*	4.82	4.72	(538, 32, 60, 299)	4.72	(538, 32, 60, 299)
	(547.9)	(7.4)	(19.8)		(89.4)	
R13-F10-C1	365800*	5.63	5.62	(717, 37, 68, 381)	5.62	(717, 37, 68, 381)
	(116.6)	(13.7)	(45.2)		(118.6)	
R13-F01-C2	150977*	1.76	1.64	(168, 24, 38, 136)	1.64	(169, 25, 38, 190)
	(90.1)	(0.7)	(2.8)		(117.6)	
R13-F05-C2	282682*	4.87	4.63	(367, 30, 49, 269)	4.63	(367, 30, 49, 273)
	(3551.8)	(11.1)	(27.8)		(73.7)	
R13-F10-C2	406790*	6.81	6.37	(457, 31, 55, 289)	6.37	(457, 31, 55, 293)
	(29695.6)	(21.2)	(16.8)		(96.4)	
R13-F01-C8	208088	3.53	3.02	(61, 12, 13, 59)	3.02	(61, 12, 13, 59)
	(t)	(7.2)	(2.9)		(62.6)	
R13-F05-C8	446997	5.47	4.84	(135, 26, 35, 142)	4.83	(136, 27, 35, 195)
	(t)	(31.3)	(11.4)		(207.8)	
R13-F10-C8	698280	5.31	4.54	(184, 31, 34, 239)	4.54	(184, 31, 34, 244)
	(t)	(54.6)	(20.8)		(85.7)	

Table 24: Computational Results Simple and Advanced Cutting-Plane-MNCI, **R13**

DESCR 20,200,100	SOL	GAP-LP	SCP-MNCI GAP	SCP-MNCI (SI,LCI,LMCI,MNCI)	ACP-MNCI GAP	ACP-MNCI (SI,LCI,LMCI,MNCI)
R14-F01-C1	403414*	0.53	0.53	(450, 26, 43, 191)	0.53	(450, 26, 43, 191)
	(495.7)	(12.9)	(15.6)		(109.1)	
R14-F05-C1	750091	5.04	5.03	(994, 33, 57, 307)	5.03	(994, 33, 57, 307)
	(t)	(123.4)	(93.7)		(234.9)	
R14-F10-C1	1078595	7.30	7.28	(1333, 36, 60, 397)	7.28	(1333, 36, 60, 397)
	(t)	(319.2)	(232.8)		(374.0)	
R14-F01-C2	437607*	0.91	0.90	(327, 18, 22, 143)	0.90	(327, 18, 22, 143)
	(1327.3)	(18.3)	(14.2)		(119.4)	
R14-F05-C2	852330	4.35	4.29	(623, 26, 30, 267)	4.29	(623, 26, 30, 267)
	(t)	(223.1)	(72.1)		(201.5)	
R14-F10-C2	1230749.7	5.19	4.96	(775, 36, 38, 311)	4.96	(775, 36, 38, 311)
	(t)	(515.8)	(107.4)		(243.2)	
R14-F01-C8	670064	2.50	2.40	(147, 14, 13, 128)	2.40	(147, 14, 13, 128)
	(t)	(121.4)	(26.9)		(142.7)	
R14-F05-C8	1643376	6.09	5.92	(287, 24, 36, 264)	5.92	(287, 24, 36, 264)
	(t)	(321.1)	(62.3)		(189.4)	
R14-F10-C8	2630332	6.38	6.20	(347, 23, 49, 164)	6.20	(347, 23, 49, 164)
	(t)	(591.3)	(63.6)		(190.9)	

Table 25: Computational Results Simple and Advanced Cutting-Plane-MNCI, **R14**

DESCR 20,200,200	SOL	GAP-LP	SCP-MNCI GAP	SCP-MNCI (SI,LCI,LMCI,MNCI)	ACP-MNCI GAP	ACP-MNCI (SI,LCI,LMCI,MNCI)
R15-F01-C1	1000787*	0.66	0.66	(708, 17, 16, 214)	0.66	(708, 17, 16, 214)
	(5384.7)	(95.5)	(81.8)		(309.9)	
R15-F05-C1	1979413	4.44	4.25	(1569, 30, 29, 378)	4.25	(1569, 30, 29, 378)
	(t)	(1923.4)	(423.1)		(681.7)	
R15-F10-C1	2949264	7.47	7.18	(2014, 33, 40, 437)	7.18	(2014, 33, 40, 437)
	(t)	(4390.3)	(576.4)		(857.8)	
R15-F01-C2	1148604	0.91	0.91	(551, 12, 9, 189)	0.91	(551, 12, 9, 189)
	(t)	(207.7)	(97.2)		(299.2)	
R15-F05-C2	2488753	3.75	3.74	(1017, 29, 30, 320)	3.74	(1017, 29, 30, 320)
	(t)	(2478.0)	(322.2)		(535.2)	
R15-F10-C2	3972667	7.36	7.33	(1301, 28, 38, 327)	7.33	(1301, 28, 38, 327)
	(t)	(9459.6)	(473.4)		(721.7)	
R15-F01-C8	2301326.6	1.10	0.94	(275, 17, 23, 67)	0.94	(275, 17, 23, 67)
	(t)	(525.9)	(96.1)		(247.2)	
R15-F05-C8	5573412.8*	0.50	0.48	(424, 24, 44, 67)	0.48	(424, 24, 44, 67)
	(16107.9)	(967.7)	(178.1)		(343.5)	
R15-F10-C8	8696932*	0.21	0.21	(562, 28, 51, 40)	0.21	(562, 28, 51, 40)
	(18930.9)	(1224.4)	(368.5)		(456.2)	

Table 26: Computational Results Simple and Advanced Cutting-Plane-MNCI, **R15**